NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER

Bethesda, Md. 20034

A GENERAL PURPOSE DATA GENERATOR FOR FINITE ELEMENT ANALYSIS

James M. McKee
Evangeline T. Marcus

Approved for public release; distribution unlimited.

COMPUTATION AND MATHEMATICS DEPARTMENT
RESEARCH AND DEVELOPMENT REPORT

April 1973

Report 4066

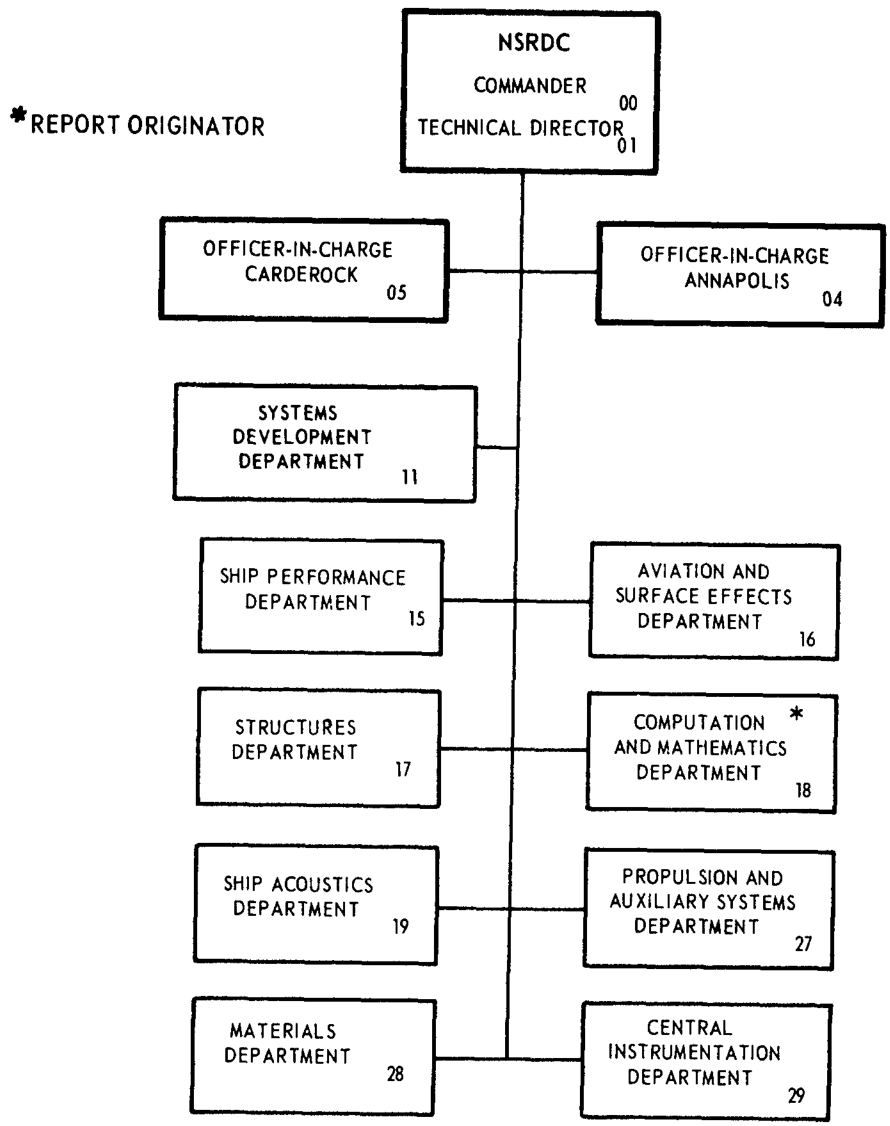
A GENERAL PURPOSE DATA GENERATOR FOR FINITE ELEMENT ANALYSIS

The Naval Ship Research and Development Center is a U. S. Navy center for laboratory effort directed at achieving improved sea and air vehicles. It was formed in March 1967 by merging the David Taylor Model Basin at Carderock, Maryland with the Marine Engineering Laboratory at Annapolis, Maryland.

Naval Ship Research and Development Center

Bethesda, Md. 20034

MAJOR NSRDC ORGANIZATIONAL COMPONENTS



DEPARTMENT OF THE NAVY
NAVAL SHIP RESEARCH AND DEVELOPMENT CENTER
Bethesda, Maryland 20034

A GENERAL PURPOSE DATA GENERATOR FOR FINITE ELEMENT ANALYSIS

by

James M. McKee
Evangeline T. Marcus

April 1973

Report 4066

TABLE OF CONTENTS

Page

Page

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

A computer program system has been developed to automate the preparation of a finite element model to be analyzed using the NASTRAN general purpose structural analysis program. Using engineering conventions and modular "building block" specifications, the program minimizes both the manual effort and the probability of an undetected error in the preparation of NASTRAN data.

This document is intended to be both a guide for the user of the program and a programmer's reference for the modification and further development of the program.

ADMINISTRATIVE INFORMATION

The work reported herein was performed as part of Task 15326, Task Area SF 53 532 106 under Work Unit number 1-1844-916 at the Naval Ship Research and Development Center.

INTRODUCTION

With the availability of large structural analysis programs based on the finite element method, the structural analyst has at his disposal a reliable tool which enables him to analyze a truly arbitrary structure to any desired accuracy. Basic to a finite element analysis is the point-by-point description of a mesh superimposed on the structure to be analyzed. As seen on the designer's blueprint, the shapes of man-made structures seldom, if ever, approach the mathematical arbitrariness attainable by this method, but instead have a topography which can be completely described by a small set of surface equations. With few exceptions today's finite element programs place the burden of translation from simple equations to a point-by-point mesh description directly on the program user. Usually, this is a manual translation which is tedious, time consuming, and highly susceptible to human errors.

NASTRAN[1] is a powerful finite element analysis program which is close to the state-of-the-art for many types of analysis* and is particularly well suited for large problems. Although it has many user convenience and data verification features, NASTRAN does require a point-by-point topographical description of structures to be analyzed.

The data generation program described in this document automatically generates a NASTRAN finite element idealization from an engineering description of the structure. In its present state the program can generate data for a specific class of structures (viz., those easily described in a cylindrical coordinate system); however, it is designed

---

1 "The NASTRAN User's Manual (Level 15)", edited by C.W. McCormick, NASA SP-222(01), June 1972.

* The types of analysis available using NASTRAN include static stress analysis, buckling analysis, natural frequency and normal mode analysis, and transient and frequency response analysis.

to be easily expanded to idealize almost any type of structure. This flexibility is achieved through the use of a data network which permits many independent data generator modules to be linked together to construct the complete idealization.

This report describes the fundamentals of structural modeling using the data generator and includes a few basic generation modules as examples.

Section 1 of the report contains information for the user of the program, describing the modeling techniques required to generate data, the facilities available for controlling the various types of data which can be generated, the conventions adopted for identifying the generated data, and detailed format specifications for the cards used to describe a structure to the generation program. Section 2 contains detailed information for those who wish to add modules or otherwise modify the program. Descriptions of the program organization, the data management philosophy, and the considerations necessary for processing the structure as a collection of independent modules have been included in this section along with the specifications for each of the program components. Section 3 is a listing of the diagnostic messages issued by the program with some additional information to clarify their meaning, and Section 4 contains selected sample problems with detailed data descriptions, samples of the generated data, printed information, and the structural plots produced by the program. Advanced modules and additional capabilities are being actively developed and supplemental reports which describe these additional capabilities will be issued periodically.

1. PROGRAM USER'S GUIDE

1.1 METHODS OF STRUCTURAL MODELING

1.1.1 Introduction

The primary purpose of the data generator is to reduce the effort required to idealize a physical structure for finite element analysis. In the process of manual idealization the engineer subdivides the structure into small, geometrically simple elements. The size of these elements is governed primarily by the following consideration: whereas the geometry and the stress distribution associated with a structure can, in general, be better approximated by smaller elements, the cost of computation increases rapidly with an increase in the number of elements. This consideration must be kept in mind when using the data generator as well.

In order to use the data generator the structure must be subdivided into regions which can accurately be described using the surface modules available in the program. As more general surface modules are included in the data generator's library, it is likely that fewer surface specifications will be required to idealize a particular structure. The user must also specify the density of the elements within each surface module so that the criteria for geometrical approximation, element assumptions, and overall problem size will be met. Finite element densities will be propagated throughout the structure into regions where user specifications have not been made. As a result of this propagation the user is required to provide only an initial specification of the element density and then to specify the regions where the density is to change. Although no word of caution will be necessary to seasoned finite element practitioners, it should be made clear that, with very few data generator commands, a finite element model can be generated which, because of its size, cannot be analyzed on any existing computer. Except for very simple problems

some judicious compromises are usually required in attempting to satisfy these criteria.

1.1.2 Reference Lines

In subdividing the structure, the boundaries between surface modules (cuts in the surfaces) will be referred to as reference lines. These reference lines provide a mechanism for linking the collection of modules which make up the complete structure and function much like the grid points (or node points) which link finite elements. The intersections of reference lines provide reference coordinates for locating modules on the structure. The reference coordinates are also used to form the grid point and element identification numbers for the data generated by adjacent modules. This convention was adopted so that the user can easily relate the generated data to the data generator specifications which produced the idealization.

It is necessary to insure that all grid points (all elements, all materials, etc.) have unique identification numbers within a particular problem. However, at intersecting surfaces within a structure, two or more structural modules could use the same reference coordinate to form data identification numbers, violating the uniqueness requirement. This problem necessitated the creation of a device which will be referred to as equivalencing of reference lines. This device permits a reference line established for geometric and module linking purposes to be treated as two or more reference lines for data identification purposes. The device also adds a certain amount of flexibility to the data generator in that it permits the inclusion of three-sided surface modules within a rectangular network.

1.1.3 Structural Modules

The data generator idealization of the region of a structure defined by a single geometric specification will be referred to as a structural module.

Each structural module is generated by a collection of data generation programs, referred to as a program module. At present there are provisions for generating four types of structural modules: surface modules, solid modules, stiffener modules, and loading condition and constraint modules.

1.1.3.1. Surface Modules. All subdivisions of a physical structure which are surfaces to be idealized using plate and shell finite elements will be referred to as surface modules. Each surface module may have either three or four edges, the shape of each edge being determined by the particular module selected. (Each surface module will obtain the geometry of its edges from adjacent modules if they have been defined in prior specifications; otherwise this edge description must be included as part of the module description.) The finite element mesh for a surface module is determined by the grid point density along its edges; for most surface modules this density can be different for each edge. Surface modules may include provisions for requesting that pressure loadings, stiffeners, and other structural features (which are not mathematical surfaces) be generated for the idealized surface. Also the user may specify that the pressure loads, stiffener properties, and the thickness properties of the finite elements are to vary in some manner over the surface.

1.1.3.2. Solid Modules. All subdivisions of a physical structure constituting regions which should be idealized using solid finite elements will be referred to as solid modules. Each solid module may have four, five, or six faces. Finite element and grid point densities within each solid module are determined by the densities specified on the edges formed by the intersection of the module faces. Certain solid modules may also be specified as hybrid modules to be used as interfaces between surface modules and pure solid modules. There are no solid or hybrid modules in the current data generator library.

1.1.3.3 Stiffener Modules. Any portions of the structure which the user desires to idealize as individual members running along a reference line will be referred to as stiffener modules. These modules are defined to extend between a starting intersection of reference lines and an ending intersection and to traverse all intersections along that path. When the path defined in this manner is not unique, the user must specify the path to be taken. As with all data generator modules one type of stiffener module may idealize a particular member using one element type while a different module may use combinations of different elements to idealize the same member. There is no restriction on the types of elements used in stiffener module idealizations as long as they are applicable to the type of analysis in which they will ultimately be used.

1.1.3.4 Loading Condition and Constraint Modules. Program modules which generate data cards for static and dynamic loadings applied along a reference line, and point loads applied at an intersection of reference lines will be referred to as loading modules. Similarly modules which generate data cards for constraints applied along a reference line (or at an intersection point) will be referred to as constraint modules. Loading condition modules and constraint modules are defined using the same reference specifications as those used for stiffener definition.

1.1.4 Modeling Procedures

The first capability developed for the data generator makes possible the idealization of quasi-axisymmetric structures defined in a cylindrical coordinate system. Because data generator modeling procedures are not dependent on the coordinate system used for structural definition, the following discussion is generally applicable to all data generator modules although it refers to cylindrical coordinates and to terms associated with structural definition in a cylindrical coordinate system.

Procedures for structural modeling with the data generator fall into two rather distinct phases: Definition of module boundaries, and

description of individual modules.

1.1.4.1 Reference Line Definition. In the definition of module boundaries the user of the program establishes the reference lines which will be required for module definition. This process is basically one of deciding how the available modules can be used to describe the structure. In most cases module boundaries will coincide with boundaries of the total structure, large stiffening members, or other natural divisions.

One approach to reference line definition is to first divide the structure transversely into sections which can be treated conveniently. The lines of division form the first set of required reference lines, called longitudinal or Z-reference lines. Figure 1.1a illustrates this longitudinal division with identification numbers assigned to the reference lines. Where branch points occur in a structure, the definition of multiple reference lines is often required to produce generated data which will be uniquely identified (these lines must later be equivalenced). Figure 1.1b illustrates one method of determining whether a multiple definition is required for Z-reference lines. With this method the user draws an arrow from the first boundary to the second boundary of each section along a profile of the structure (the direction specified by the arrow also indicates the order in which data will be specified in module definition). Each terminal along this profile will require the definition of as many reference lines as there are arrows emanating from that terminal. Thus the approach used in Figure 1.1b requires two Z-reference lines at the circled terminal while the approach used in Figure 1.1c requires no multiple definitions. Multiple definitions sometimes simplify input specification and are permitted for any reference line in the structure, whether or not such a definition is required for uniqueness.

Once the structure has been divided transversely to define Z-reference lines, azimuthal or A-reference lines can be established by dividing the structure azimuthally as shown in Figure 1.2a. The

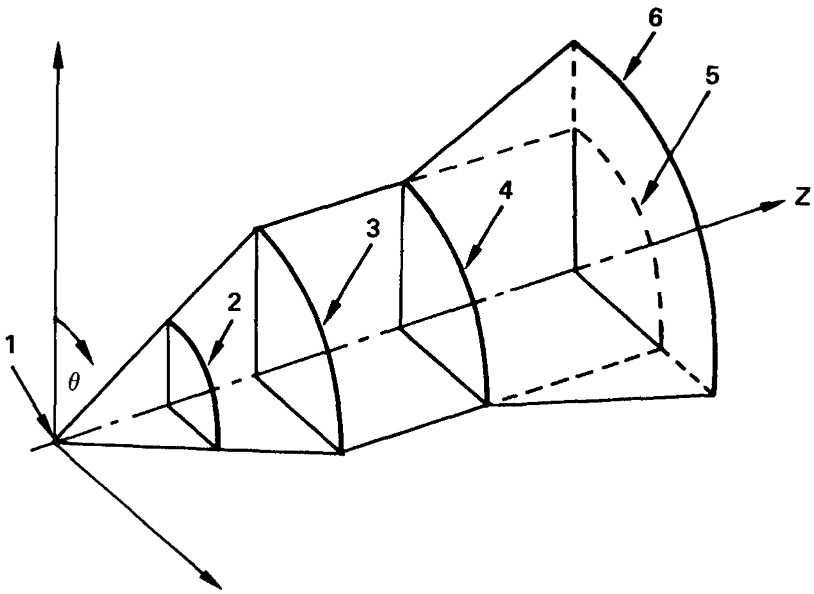


Figure 1.1a – Longitudinal Reference Lines

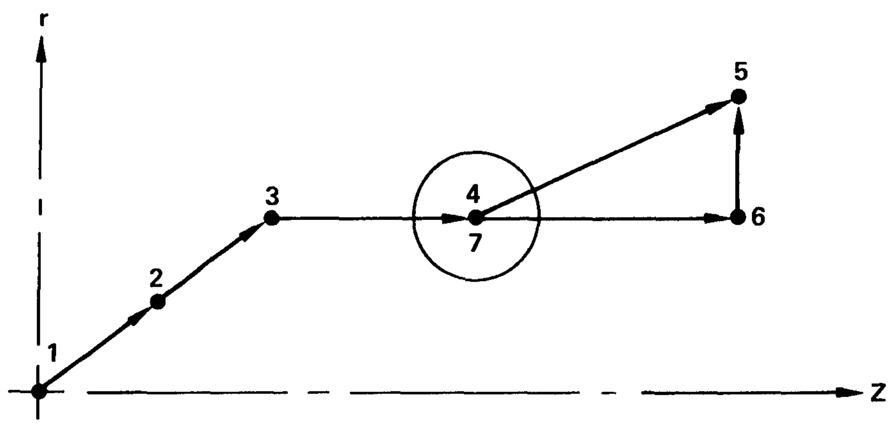


Figure 1.1b – Example Requiring Multiple Definition

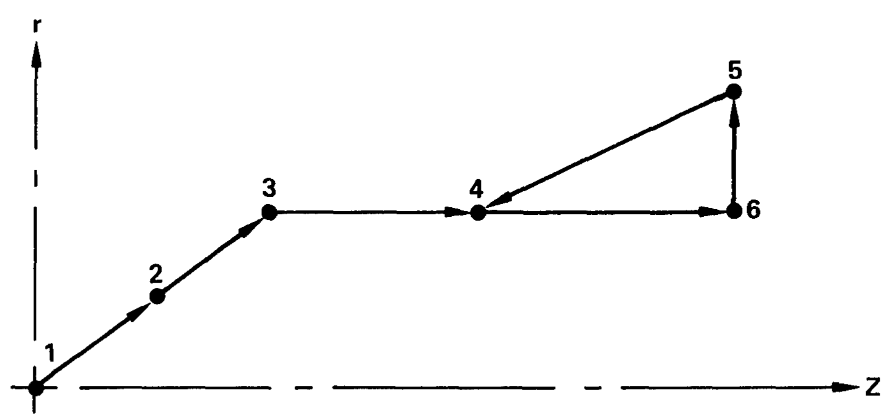


Figure 1.1c – Example Not Requiring Multiple Definition

Figure 1.1 – Longitudinal Reference Line Definition

procedure is identical to the one used for longitudinal reference lines, including the establishment of multiple reference lines. Figure 1.2b illustrates the A-reference line division and numbering of a structure with two branch points, one of which requires multiple reference line definition (circled terminal) and one which does not (terminal numbered 14).

1.1.4.2 Description of Structural Modules. Once reference lines have been established, individual modules can be described. Because each different type of structural module is a data generator with its own specifications, there are few generalizations one can make about module description. In Section 2.1.2 some proposed standard data format conventions are presented. Although these standards were written as a guide to those programming data generation modules, they may also be of some help to the user in preparing data for the program.

1.1.5 Special Topologies and Data Equivalencing

Conventions that have been established in writing the data generation program, such as using the reference line identification numbers to produce unique identification numbers for generated data, and storing boundary information as if the module boundaries formed rectangular lattices on the structure, often conflict with the topology of the structure to be modeled. The problem of generated data with duplicate identification numbers can be seen in Figure 1.2. If one chooses to associate the reference line number with the module following it as one proceeds around the arc in a particular direction, say clockwise, it is not possible to associate a unique reference line number with every module since there are 16 modules and only 15 reference lines.

A device, referred to as equivalencing, was introduced to resolve these conflicts. Equivalencing changes pointers in the data management system so that a data group can be referenced by more than one identifier. These changes can be made either globally, for all usage of a reference

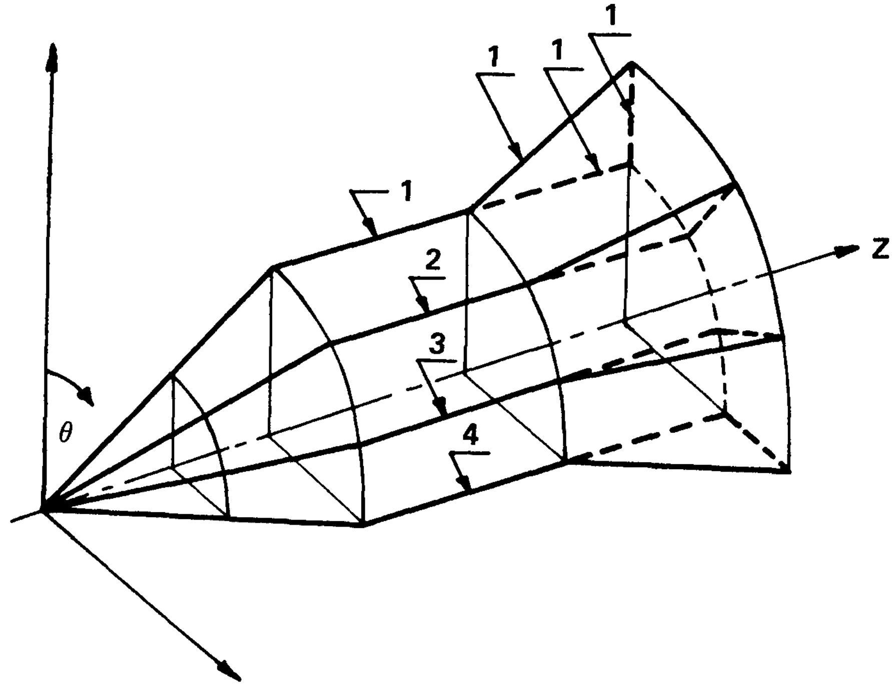


Figure 1.2a – Azimuthal Reference Lines

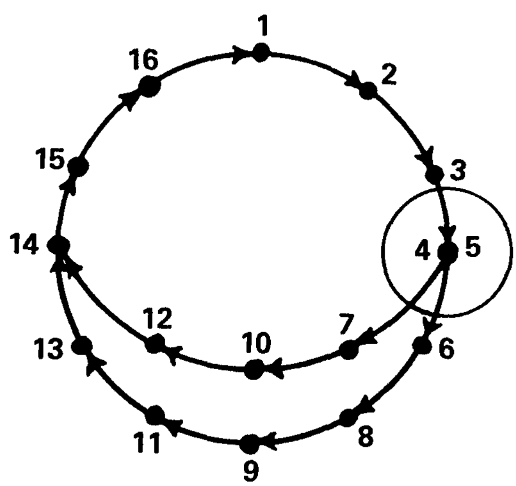


Figure 1.2b – Example Requiring Multiple Definition

Figure 1.2 – Azimuthal Reference Line Definition

line number or individually, for data stored at the intersection of two data lines, or between any two data groups. Generation modules may also invoke equivalences to enforce their special topology. For example, the CONEND module invokes equivalences between the intersection data at two corners of the rectangular lattice, creating a triangular boundary. The user may employ similar techniques to create box structures from plate modules.

Global equivalences are invoked by listing pairs of reference line numbers on ZEQU bulk data cards for Z-reference lines and on AEQU cards for A-reference lines. Equivalences between particular data groups can be invoked with IEQU and GEQU bulk data cards. (See Section 1.4.3 for a complete description of these cards.)

1.2 PROGRAM SETUP

1.2.1 Card Deck Format

The computer run deck for the data generator consists of two parts: the Execution Control Deck and Bulk Data Deck. The Execution Control Deck is used to select the type of data to be generated and to control the printed and graphical output of the program. The Bulk Data Deck is used to describe the geometry of the structure to be modeled. The Execution Control Deck begins with an ID card and is terminated with a BEGIN BULK card. The Bulk Data Deck follows the Execution Control Deck and continues through the ENDDATA card.

All cards preceding the ID card, the ID card, the BEGIN BULK card, and the ENDDATA card are inserted into corresponding positions in the generated NASTRAN deck file. Other data cards may be passed from both the Execution Control Deck and the Bulk Data Deck directly to the generated data file by terminating the data generator cards with a $END card. Execution Control cards between the $END card and the BEGIN BULK card will be inserted in the Executive/Case Control portion of the NASTRAN deck; the data generator Bulk Data cards

between a $END and the ENDDATA card will be inserted as the first cards in the Bulk Data portion of the generated NASTRAN deck. All cards following the ENDDATA card will be ignored by the program.

The delimiting cards used by the program have the following format:

ID A1, A2 Required

A1 and A2 are any alphanumeric fields chosen by the user. These fields are optional to the data generator, but are required by NASTRAN.

BEGIN BULK Required

The two distinct words must appear on the card.

ENDDATA Required

ENDDATA must be one continuous word beginning in the first column of the data card.

$END Optional

$END must be one continuous word beginning in the first column of the data card.

The Execution Control Deck and the Bulk Data Deck are described in detail in Sections 1.2.2 and 1.4, respectively.

1.2.2 Execution Control Cards

Data cards in the program run deck between the ID card and the first $END (or BEGIN BULK card if there is no $END card) will be interpreted as execution control cards. All execution control cards have default specifications and thus the only control cards which are mandatory are the ID card and the BEGIN BULK card. Warning messages will be issued when cards cannot be recognized and when cards do not conform to the prescribed syntax. Many of the options governed by control cards depend on implementation by the writers of the various data generation modules. The names of cards with module dependent options are flagged with an asterisk in the following card descriptions, indicating that the module descriptions (Section 1.3.2) should be consulted for information on their applicability.

All cards read by the program are listed and copied to the generated data file. All execution control cards, except comment cards, will have a dollar sign appended at the left, making them comments to NASTRAN. All cards preceding the ID card and between the $END card and the BEGIN BULK cards will be listed and copied to the generated data file without alteration, thus permitting the user to transmit NASTRAN control cards directly to the generated data file.

Comment cards are those cards which are either totally blank or have a dollar sign as the first non-blank character.

Examples of Execution Control Decks are shown in Figures 1.3 and 1.4. The second example illustrates the type of specifications which would be required for a combination data generation and NASTRAN analysis run. The cards following the $END will not be interpreted by the program even though they may have a format which is similar to the data generator (e.g., SOL, TITLE, PLOTID cards).

```
ID MYNAME, CODE 1234
$ EXECUTION CONTROL DECK FOR STAND-ALONE OPERATION
DUMP = NONE
LINES = 38
SOL = 3
TITLE = DATA GENERATOR  TEST NO. 101
PLOTID = MYNAME,  CODE 1234,  EXT 51234
BEGIN BULK
```

Figure 1.3 Example of Execution Control Deck for Stand-Alone Data Generator Operation

```
ID J. SMITH, PROB-1Q4
$ EXECUTION CONTROL DECK FOR DATA GENERATOR/NASTRAN
SOL = 2
PUNCH = YES
TITLE = FRAME 17 MODIFICATION ANALYSIS - DATA GEN.
NASTRAN = YES
$END
APP  DISP
SOL  2,0
TIME  100
CHKPNT  YES
CEND
TITLE = FRAME 17 MODIFICATION ANALYSIS
PLOTID = J.  S M I T H,  C O D E  1 5 5 5
    .
    .
    .
BEGIN BULK
```

Figure 1.4 Example of Execution Control Deck for Data Generation/NASTRAN Operation

The format of the execution control cards is free-field. In presenting general formats for each card embodying all options, the following conventions are used:

(1) Upper-case letters must be punched as shown.

(2) Lower-case letters indicate that a substitution is to be made.

(3) Braces { } indicate that a choice of contents is mandatory.

(4) Underlined options or values are the default values.

(5) Physical card consists of information punched in columns 1 through 72 of a card. All data generator control cards are limited to a single physical card.

(6) Card names with asterisks indicate module dependent options.

Execution Control Card - <u>CONNECT*</u>

Description: This card controls the generation of connection cards.

Format:

$$\text{CONNECT} = \left\{ \begin{array}{l} \underline{\text{YES}} \\ \text{NO} \end{array} \right\}$$

Option	Meaning
YES	Connection cards are to be included in the generated data file.
NO	No connection cards are to be included in the generated data file.

Execution Control Card - <u>DUMP</u>

Description: This card controls the printing of various program data storage area dumps.

Format:

$$\text{DUMP} = \left\{ \begin{array}{l} \text{NONE} \\ \text{DIAG(n)} \\ \text{NOFORM(n)} \\ \text{FATAL(n)} \\ \underline{\text{FATAL(2000)}} \end{array} \right\}$$

Option	Meaning
NONE	No data storage dumps printed.
DIAG(n)	All data storage area dumps are printed with a maximum of n words printed per storage area. Chained storage areas are interpretively formatted so that only the stored information is printed (i.e., all chain pointers, etc. are excluded).
NOFORM(n)	All data storage area dumps are printed with a maximum of n words printed per storage area. The areas are printed as a block with no interpretive formatting.
FATAL(n)	A data storage area dump is printed with a maximum of n words per storage area following a fatal error. Dumps will have the same format as those produced by a DIAG request.

<u>Remark:</u> The word count, n, must be greater than zero.

Execution Control Card - <u>ERRORS</u>

Description: With this card the user may specify the number of level 2 errors which will be permitted in a given phase of processing before the run is terminated.

Format:

$$\text{ERRORS} = \begin{Bmatrix} n \\ \underline{50} \end{Bmatrix}$$

Option	Meaning
n	The number of errors permitted (≥ 0)

Execution Control Card - <u>FATAL</u>

Description: With this card the user may declare which errors detected by the program will be considered fatal to execution.

Format:

$$\text{FATAL} = \begin{Bmatrix} \text{ALL} \\ \underline{\text{NORMAL}} \\ \text{NONE} \end{Bmatrix}$$

Option	Meaning
ALL	Any error detected will cause termination of the program.
NORMAL	Errors will be considered fatal only when continued execution would clearly result in nonsensical results.
NONE	Execution will be permitted to continue following all errors.

Execution Control Card - FORCE*

Description: This card controls the generation of NASTRAN FORCE cards.

Format:

$$\text{FORCE} = \begin{Bmatrix} \underline{\text{YES}} \\ \text{NO} \end{Bmatrix}$$

Option	Meaning
YES	NASTRAN FORCE cards are to be included in the generated data file.
NO	No FORCE cards are to be included in the generated data file.

Execution Control Card - LINES

Description: This card permits the user to specify the number of lines to be printed on each page of the generated listing.

Format:

$$\text{LINES} = \begin{Bmatrix} n \\ \underline{55} \end{Bmatrix}$$

Option	Meaning
n	The number of lines per printed page (> 10).

Execution Control Card - MAT1*

Description: This card controls the generation of NASTRAN MAT1 cards with default material properties for all materials not explicitly defined.

Format:

$$\text{MAT} = \left\{ \begin{array}{c} \underline{\text{YES}} \\ \text{NO} \end{array} \right\}$$

Option	Meaning
YES	NASTRAN MAT1 cards are to be included in the generated data file.
NO	No MAT1 cards are to be included in the generated data file.

Remark: See module descriptions (Section 1.3.2) for default values.

Execution Control Card - NASTRAN

Description: This card indicates that a NASTRAN analysis is to immediately follow this Data Generator application.

Format:

$$\text{NASTRAN} = \left\{ \begin{array}{c} \text{YES} \\ \underline{\text{NO}} \end{array} \right\}$$

Option	Meaning
YES	NASTRAN analysis follows; only one set of data will be generated in this run.
NO	No analysis run follows; multiple sets of data will be processed, continuing until all sets have been completed.

Execution Control Card - PLOAD*

Description: This card controls the generation of NASTRAN PLOAD cards.

Format:

$$\text{PLOAD} = \left\{ \begin{array}{c} \underline{\text{YES}} \\ \text{NO} \end{array} \right\}$$

Option	Meaning
YES	NASTRAN PLOAD cards are to be included in the generated data file.
NO	No PLOAD cards are to be included in the generated data file.

Execution Control Card - PLOTID

Description: This card permits the user to request that structural plots be made of the generated model and to identify the plots produced to the plotter operator.

Format:

PLOTID = {name, CODE code, EXT ext}

Parameter	Meaning
name	User's name; up to eight characters with no embedded blanks.
code	User's organizational code; up to four characters with no embedded blanks.
ext.	User's telephone extension; up to five characters with no embedded blanks.

Remarks: (1) If no PLOTID card is present, no structural plots will be generated.

(2) A tape must be mounted to receive the generated plotting information (see Section 2.1.4).

Execution Control Card - <u>PRINT*</u>

Description: This card controls the printing of data generation information messages.

Format:

$$\text{PRINT} = \left\{ \begin{matrix} \text{MAX} \\ \underline{\text{MIN}} \end{matrix} \right\}$$

Option	Meaning
MAX	All messages generated will be printed.
MIN	Messages which do not directly concern the program user are not printed (these are undocumented messages which programmers have included to facilitate program testing).

Execution Control Card - <u>PUNCH</u>

Description: This card controls the punching of the generated data file at the end of the application.

Format:

$$\text{PUNCH} = \left\{ \begin{matrix} \underline{\text{NO}} \\ \text{YES} \end{matrix} \right\}$$

Option	Meaning
NO	No generated data will be punched; the data will be written only on the generated data file.
YES	Generated data will be punched on cards as well as being placed on the generated data file.

Execution Control Card - SOL*

Description: This card indicates which NASTRAN Rigid Format will be used to analyze the generated model.

Format:

$$\text{SOL} = \left\{ \begin{matrix} n \\ \underline{1} \end{matrix} \right\}$$

Option	Meaning
n	NASTRAN Rigid Format number n will be used to analyze the generated model. ($0 < n \leq 12$, n is an integer).

Execution Control Card - SPC*

Description: This card controls the generation of NASTRAN SPC cards.

Format:

$$\text{SPC} = \left\{ \begin{matrix} \underline{\text{YES}} \\ \text{NO} \end{matrix} \right\}$$

Option	Meaning
YES	NASTRAN SPC cards are to be included in the generated data file.
NO	No SPC cards are to be included in the generated data file.

Execution Control Card - TITLE

Description: This card permits the user to supply a title to be printed on each page of the generated listing.

Format: TITLE = any text

Remarks: (1) The text on the TITLE card will be printed at the top of each page along with the current data and page number.

(2) If no TITLE card is included, only the date and page number will be printed.

1.3 GENERATING DATA

1.3.1 Data Organization and Identification

NASTRAN restrictions limit the grid point and element identification numbers generated by the program to seven digits in length. The data generator could conveniently construct grid point identification numbers by associating two digits with each reference line at an intersection and then allowing two digits for each coordinate direction to locate the gridpoint within each module. This would permit a maximum of 100 gridpoints along any edge of a module and permit 99 reference lines in each of the two directions. The range of grid point numbers generated in this way would be sufficient for most models and the element identification numbers could be assigned in a similar manner with a range sufficient for a module of any solvable size. However, this approach requires eight-digit identification numbers.

The method used is basically a six-digit scheme allowing one digit for each reference line and two digits for each coordinate direction within a module. With the available seventh digit used as an overflow digit to permit an increased number of reference lines, 39 reference lines in one coordinate direction and 19 in the second direction can be accommodated. Table 1.1 gives the correspondence between the seventh digit in the reference line number and the range of Z- and A-reference line numbers. The grid point identification number $\underline{78}01\underline{6}02$, for example, lies within a module which has its upper left-hand grid point at the intersection of Z-reference line 38 and A-reference line 16. The seventh (leftmost) digit indicates that the Z-reference line is in the range 30 to 39 and that the A-reference line is in the range 10-19. The sixth and third digits complete the specification of the Z-reference line number and the A-reference line number, respectively. Within a module, excluding its boundary, the first and second digits and the fourth and fifth digits, each pair taken as an integer, are always non-zero.

TABLE 1.1 - CORRESPONDENCE BETWEEN REFERENCE LINE NUMBERS AND THE SEVENTH (LEFTMOST) DIGIT OF GRID POINT IDENTIFICATION NUMBERS

Seventh Digit	ZRL	ARL
blank	1-9	1-9
1	10-19	1-9
2	20-29	1-9
3	30-39	1-9
4	1-9	10-19
5	10-19	10-19
6	20-29	10-19
7	30-39	10-19

Along a Z-reference line, away from an intersection, the first and second digits are zero and the fourth and fifth digits are non-zero. Along an A-reference line, the fourth and fifth digits are zero and the first and second digits are non-zero.

Element identification numbers generated by a module also follow the seven-digit pattern and have the same reference line digits as the grid point in the upper left corner of the module. Identification numbers generated within a module vary with the module used and are discussed in Section 1.3.2. The reference line numbers used in forming the grid point and element identification numbers are internal numbers and not necessarily the external number which the user has supplied. The internal number assigned to a reference line is its position in the sequence of reference lines in that direction as they are encountered on module specification cards. The only restrictions on the external numbers are that they be greater than zero and that the number of reference lines does not exceed 39 for Z-reference lines and 19 for A-reference lines.

This numbering scheme was designed primarily so that the user can easily identify the module that generated the data. Use of the gridpoint numbering, as generated, will usually result in structural matrices with abnormally large bandwidths, and hence, long running times in the NASTRAN analysis. The BANDIT program,[2] also developed under this project, provides an efficient gridpoint renumbering, is simple to use, and is generally available to NASTRAN users.

1.3.2 Module Descriptions

Unless specifically stated otherwise any consistent set of units may be used to define module geometry and properties. Angles must be specified in degrees unless otherwise noted. Any location data (r, θ, z) which have been defined in a previous module need not be redefined for subsequent modules. If the location of a point is specified in two or more modules, the first specification encountered will take precedence over all subsequent specifications in the event of a data conflict.

---

[2] Everstine, Gordon C., "The BANDIT Computer Program for the Reduction of Matrix Bandwidth for NASTRAN," Naval Ship Research and Development Center Report 3827 (March 1972).

Module Name: CONE

Function: To generate a finite element model of a sector of a frustum of a general (non-circular) cone using the quadrilateral and/or triangular elements QUAD2 and TRIA2.

Geometric Considerations:

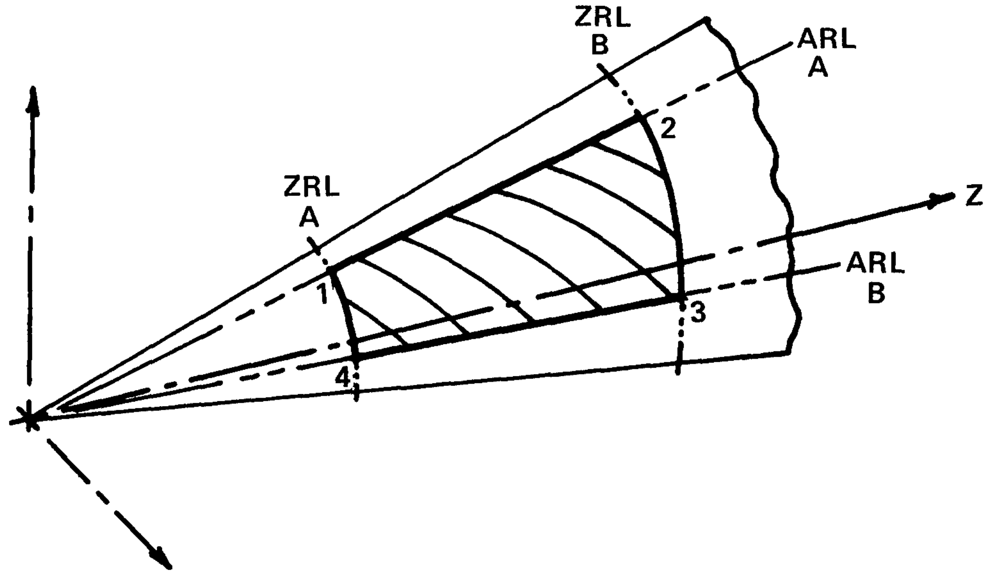


Figure 1.5 – CONE Module

The surface to be modeled with the CONE module is assumed to be describable by a linear equation in terms of axial position and azimuthal angle in a cylindrical coordinate system. The edges of the module at ZRL A and ZRL B as shown in Figure 1.5 are assumed to lie in a plane perpendicular to the Z-axis. Similarly, the edges at ARL A and ARL B each lie in a plane of constant azimuthal angle. If the region to be modeled requires a more complex geometrical description, either a different generating module should be used or the

region should be subdivided into smaller regions which can be approximated by the CONE module.

The thickness of the material and the pressure loading on the module are both assumed to vary linearly with axial position and azimuthal angle. Only one type of material is permitted for the entire module; however, the material may be easily specified as being anisotropic since each element material reference system has its major axis oriented so that it always lies in an r, z-plane, being parallel to the z-axis whenever the element also lies in an r, z-plane. The user's choice of the NASTRAN material properties card governs whether isotropic or anisotropic material properties will be used.

Type of Model Generated: The surface of the model is approximated by a mesh of the planar QUAD2 and TRIA2 elements, with the grid points of the elements lying on the surface described by the user. The generated model is thus a polyhedron which always lies on the concave side of the structure being modeled. The location of grid points along the boundaries is calculated by linear interpolation between the corner points. The location of interior points is calculated by averaging the values obtained by interpolating first between Z-reference line values and then A-reference line values. The same type of interpolation is used to calculate element thicknesses and pressure loads for the module. Because each QUAD2 and TRIA2 element is assumed to have uniform thickness and a uniform pressure load applied to its surface, the interpolation is made to the centroid of the elements for these quantities, rather than to a grid point.

The mesh density for the module is controlled by the number of grid points (or divisions) along the edges of the module. A mild restriction is made on the relationship between the number of divisions on the various edges of the module (see Section 1.4.3) in order to simplify the mesh variation within a module and to maintain acceptable element geometry. QUAD2 elements will be used for all modeling except in regions where

mesh variation requires triangular elements and in those regions TRIA2 elements will be used. The modeling method will concentrate triangular elements in regions of higher mesh density.

For reliable finite element results the generated triangular elements should be close to equilateral and the quadrilateral elements should be nearly square. If the height-to-base ratio of any element does not fall in the range ($\frac{1}{2}$, 2), the elements should be considered to have "bad" geometry. The program attempts to generate elements which have acceptable shapes, but this process is very dependent on the dimensions of the module to be generated and on the mesh density at its boundary. The ultimate responsibility for the acceptability of the elements has been left to the user.

Within the cone module (see Figure 1.5) grid point numbering starts at the top and proceeds from left to right, then from top to bottom, using the 7-digit numbering convention described in Section 1.3.1. The sixth digit contains the number of ZRL-A bounding the CONE module on the left and the fourth and fifth digits contain the count, along an axial line, of the nodes from ZRL-A. The third digit contains the number of ARL-A bounding the section at the top and the remaining two digits contain the count, along an azimuthal line, of the nodes from ARL-A. This numbering convention applies to CONE modules with rectangular meshes as well as to those with triangular meshes.

The ID of the grid point at the top left of the element is used as the element identification of quadrilaterials. The identification of triangular elements is obtained by starting with the node number of the top left grid point and numbering the triangles in the Z-direction, incrementing the fourth digit of each grid point by one. Figure 1.6 illustrates the grid point and element numbering for this module.

The idealization generated by the CONE module will be constructed using NASTRAN's CQUAD2, PQUAD2, CTRIA2, PTRIA2, GRID, and PLOAD bulk data cards. These cards will be sorted by type (connection,

property and load, and GRID) in the generated data deck. For a given type the cards will appear in the order generated.

This module does not use any of the optional output control parameters. As data cards are generated, comment cards are inserted which identify the module by the external numbers of the reference lines on its boundary.

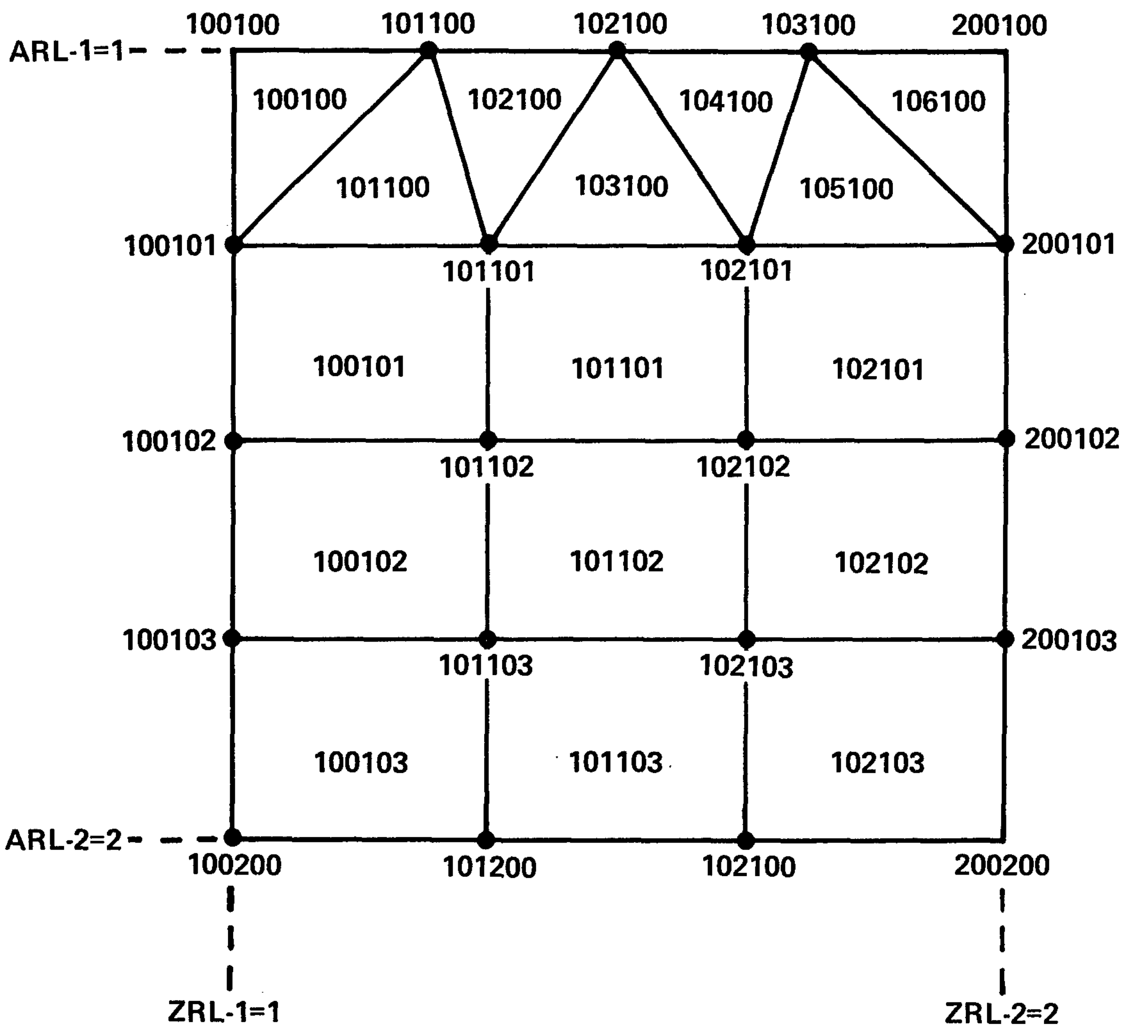


Figure 1.6 – Grid Point and Element Numbering Generated by the CONE Module

Module Name: CONEND and CONENDR

Function: To generate a finite element model of a cone-shaped end module using the quadrilateral and/or triangular elements QUAD2 and TRIA2.

Geometric Considerations:

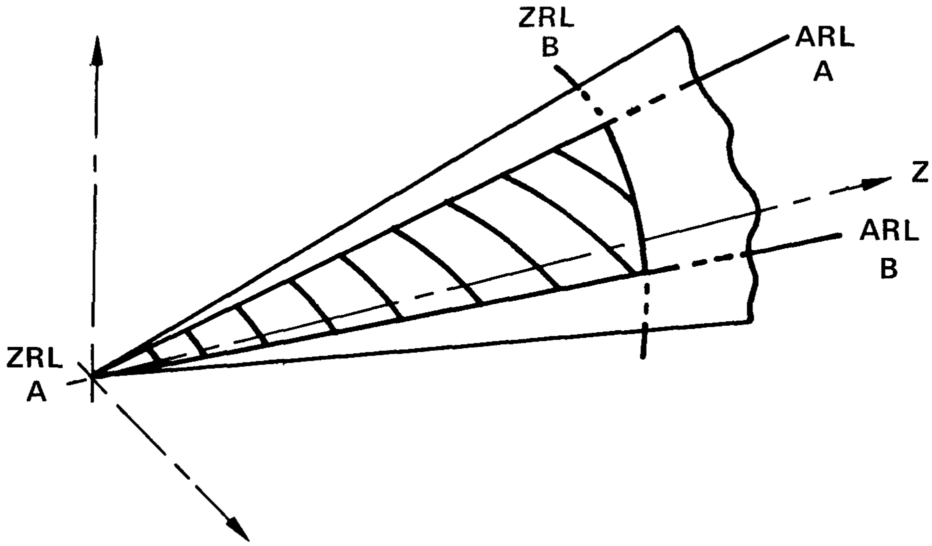


Figure 1.7 – CONEND Module

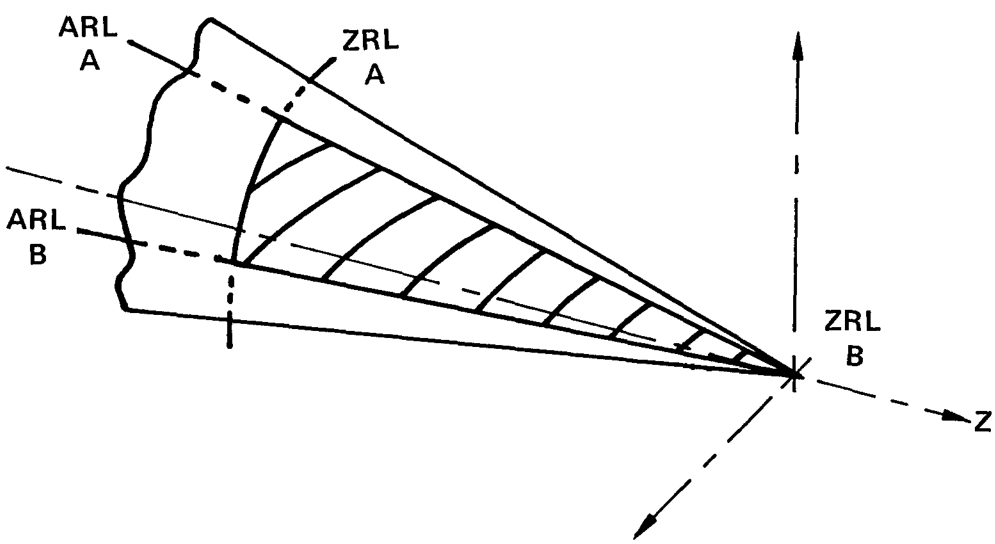


Figure 1.8 – CONENDR Module

The surface to be modeled with the CONEND or CONENDR module is assumed to be describable by a linear equation in terms of axial position and azimuthal angle in a cylindrical coordinate system. The edges of the modules at ZRL-A and ZRL-B as shown in Figure 1.7 for the CONEND and in Figure 1.8 for the CONENDR are assumed to lie in a plane perpendicular to the Z-axis. Similarly the edges at ARL-A and ARL-B each lie in a plane of constant azimuthal angle. The only difference between a CONEND and a CONENDR module is the location of the vertex with respect to the order in which the various parameters are generated and numbered. If the region to be modeled requires a more complex geometrical description, a different generating module should be used or the region should be subdivided into smaller regions which can be approximated by the CONEND or CONENDR and CONE modules (see Section 1.3.2).

The thickness of the material and the pressure loading on the module are both assumed to vary linearly with axial position and azimuthal angle. Only one type of material is permitted for the entire module; however, the material may be anisotropic as well as isotropic since each element material reference system has its major axis oriented so that it always lies in the first quadrant of an r, z-plane, being parallel to the z-axis whenever the element also lies in an r, z-plane.

Type of Model Generated: The surface of the model is approximated by a mesh of the planar TRIA2 and possibly QUADS elements with the grid points of the elements lying on the surface described by the user. The generated model is thus a polyhedron which always lies on the concave side of the structure being modeled. The location of grid points along the boundaries is calculated by linear interpolation between the corner grid points. The location of interior points is calculated by averaging the values obtained by interpolating between the A-reference line values. The same type of interpolation is used to calculate element thicknesses and pressure loads for the module, and since each TRIA2 and QUAD2 elements is assumed to have uniform thickness and a uniform

pressure load applied to its surface, the interpolation is made to the centroid of the elements for these quantities rather than to the grid points.

The mesh density for the module is controlled by the number of grid points (or divisions) along the edges of the module. A mild restriction is made on the relationship between the number of divisions on the various edges of the module (see Section 1.4.3) in order to simplify the mesh variation within a module and to maintain acceptable element geometry. TRIA2 elements will be used for all modeling except in regions where mesh variation requires quadrilateral elements and there QUAD2 elements will be used. The modeling method will concentrate quadrilateral elements in regions of lower mesh density.

For reliable results the generated triangular elements should be close to equilateral, and the quadrilateral elements should be nearly square. If the height-to-base ratio of any element does not fall in the range $(\frac{1}{2}, 2)$, the elements should be considered to have "bad" geometry. The program attempts to generate elements which have "good" shape; however, this process is very dependent on the dimensions of the module to be generated and on the mesh density at the boundary. The ultimate responsibility for the acceptability of the elements has been left to the user.

For CONEND and CONENDR modules, the grid points are numbered starting at the top left of the module and proceeding from top to bottom, working from left to right. The identification of the triangular elements is obtained by taking the node number of the top left grid point of each column of triangular elements and numbering the triangles in the theta-direction, incrementing the rightmost digit by one. Figures 1.9 and 1.10 illustrate the grid point and element numbering for the CONEND and CONENDR modules respectively.

These modules do not use any of the optional output control parameters. As data cards are generated, comment cards are inserted which identify the module by the external number of the reference lines on its boundary.

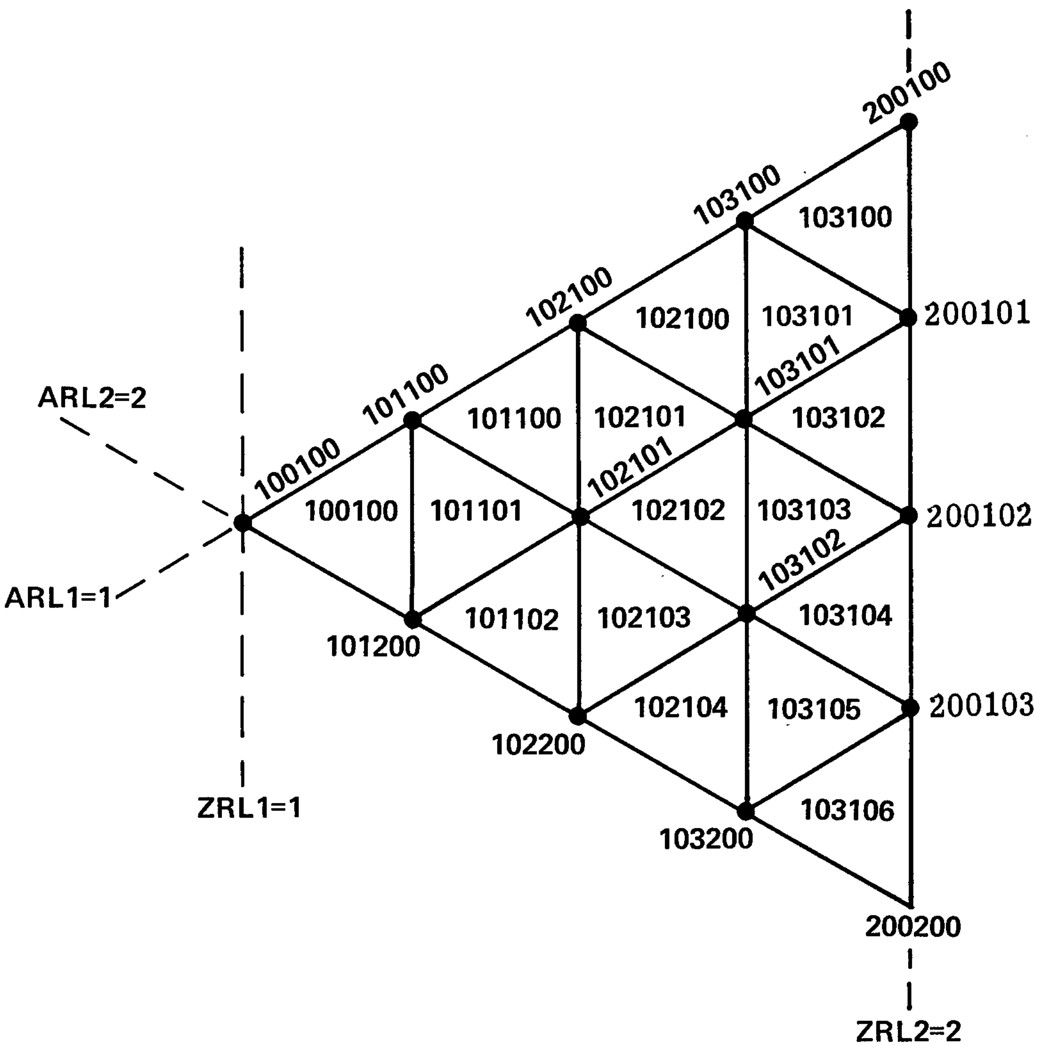


Figure 1.9 – Grid Point and Element Numbering Generated
by the CONEND Module

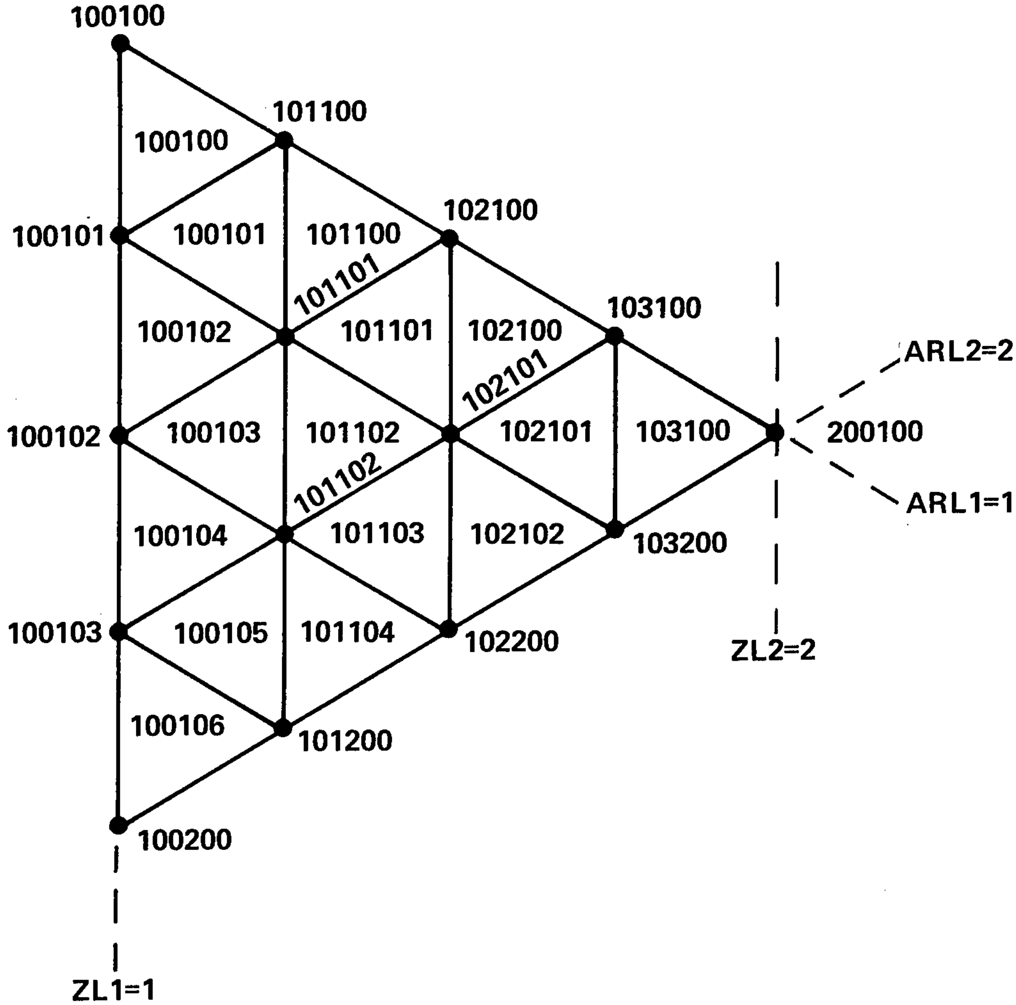


Figure 1.10 – Grid Point and Element Numbering Generated by the CONENDR Module

1.3.3 Graphical Output

The data generator will optionally produce microfilm plots of the generated structure. If a PLOTID card is included in the execution control deck, the program will produce four plots, including a perspective view and three orthogonal views from a position normal to each of the three basic coordinate planes of the structure. The graphic output will serve as a check on the geometry of the output data deck generated by the program. Structural plots for each of the sample problems are included in Section 4.

1.4 STRUCTURAL DATA SPECIFICATIONS

1.4.1 Bulk Data Card Format

The data card format is variable to the extent that any quantity except the mnemonic can be punched anywhere within a specified 8-column field. Each bulk data card consists of ten 8-column fields as indicated in the following diagram:

1	2	3	4	5	6	7	8	9	10
CONE	1	3	2	6	90.		11.25	20.0	+C01

The mnemonic is punched in field 1 beginning in column 1. Fields 2-9 are for data items. The only limitations in data items are that they must lie completely within the designated field, have no imbedded blanks, and must be integer or real numbers, or completely blank. The program will convert all numbers so that they satisfy the type requirements of the various modules.

On continuation cards, field 10 is used in conjunction with field 1 of the continuation card as an identifier and hence must contain a unique entry. The continuation card contains the symbol + in column 1 followed by the same characters that appear in columns 74-80 of field 10 of the card that is being continued.

1.4.2 Order of Bulk Data Cards

Bulk data cards containing structural module specifications must be arranged in the order in which the various components of the idealization are to be generated; otherwise, geometric information which is passed from one module to another, may not be correct. Other types of bulk data cards may be included in any desired order. Continuation cards must immediately follow their parents; however, field 10 may be non-blank even if no continuation card follows. Uniqueness of field 1 on continuation cards is not required.

1.4.3 Data Card Descriptions

The following pages contain the data format specifications for the various bulk data cards arranged in alphabetical order by card name. Table 1.2 contains a summary of the functions of the bulk data cards, arranged by card type.

TABLE 1.2 - SUMMARY OF BULK DATA CARDS

The following data input cards are available to the user:

Card Type	Card Name	Function	Page
Surface Module	CONE	To generate a cone-shaped module; described as a sector of a frustum of a general cone.	1.37
Surface Module	CONEND	To generate a cone-shaped end module; data generated from apex to base.	1.41
Surface Module	CONENDR	To generate a cone-shaped end module; data generated from base to apex.	1.44
Equivalence	AEQU	To specify the equivalence of A-reference lines.	1.36
Equivalence	GEQU	To specify the equivalence of general data quantities.	1.47
Equivalence	IEQU	To specify the equivalence of intersections of reference lines.	1.49
Equivalence	ZEQU	To specify the equivalence of Z-reference lines.	1.50

Input Data Card AEQU A-Reference Line Equivalence

Description: Defines Equivalence between A-reference lines.

Format and example:

1	2	3	4	5	6	7	8	9	10
AEQU	ARLD1	ARLI1	ARLD2	ARLI2	ARLD3	ARLI3	etc.		
AEQU	4	1	5	2					

Field	Contents
ARLDi	Dependent A-reference line number--one that will be equivalenced to ARLIi (integer > 0)
ARLIi	Independent A-reference line number--one that will be used for grid point numbering and is equivalenced to ARLDi (integer > 0)

Remarks:

1. Maximum number of A-reference lines is 19.
2. See Section 1.1.5 for a discussion of equivalencing.
3. The user must insure that equivalence specifications are not circular. There are no other restrictions regarding which reference lines may or may not be equivalenced.

Input Data Card <u>CONE</u> Cone Shaped Module

Description: Defines a module to generate data for a region which can be described as a sector of a frustum of a cone (Figure 1.11).

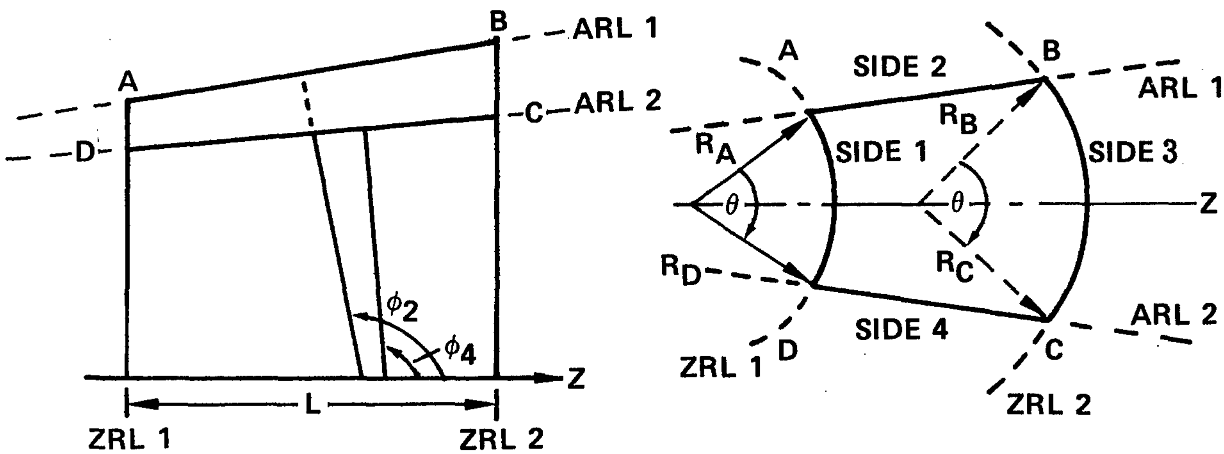


Figure 1.11 – Geometry of the CONE Module

Format and example:

1	2	3	4	5	6	7	8	9	10
CONE	ZRL1	ARL1	ZRL2	ARL2	ϕ_2	ϕ_4	L	θ	+IDENT1
CONE	1	2	5	4	105.		10.	45.	+ C101

1	2	3	4	5	6	7	8	9	10
+IDENT1	R_A	TH_A	P_A	DIV1	M				+IDENT2
+C101	1.0	1.0	1.0	6	1				+ C102

1	2	3	4	5	6	7	8	9	10
+IDENT2	R_B	TH_B	P_B	DIV2					+IDENT3
+C102		1.0	1.0	4					+ C103

1	2	3	4	5	6	7	8	9	10
+IDENT3	R_C	TH_C	P_C	DIV3					+IDENT4
+C103									+C104

1	2	3	4	5	6	7	8	9	10
+IDENT4	R_D	TH_D	P_D	DIV4					
+C104	1.5								

Field	Contents
ZRL1	Z-reference line number bounding the module on the left (side 1, integer > 0)
ARL1	A-reference line number bounding the module at the top (side 2, integer > 0)
ZRL2	Z-reference line number bounding the module on the right (side 3, integer > 0)
ARL2	A-reference line number bounding the module on the bottom (side 4, integer > 0)
ϕ_2	Angle between the positive Z-direction and the normal to the surface along ARL1; counterclockwise is positive (real, degrees)
ϕ_4	Angle between the positive Z-direction and the normal to the surface along ARL2; counterclockwise is positive (real, degrees)
L	Length of the projection of the module on the Z-axis (real ≥ 0)
θ	Angular width of the module (real, degrees)
R_A, R_B, R_C, R_D	Radii at corners A, B, C, and D (real)
TH_A, TH_B, TH_C, TH_D	Thicknesses at corners A, B, C, and D (real)
P_A, P_B, P_C, P_D	Pressures at corners A, B, C, and D (real)
DIV1, DIV2, DIV3, DIV4	Numbers of division along sides 1, 2, 3, and 4 (integer ≥ 1)
M	Material identification number (integer ≥ 1)

Remarks:

1. The two Z-reference lines must have different numbers and may not be reference lines which have been equivalenced.

2. The two A-reference lines must have different numbers and may not be reference lines which have been equivalenced.

3. If R_B has not been defined, default is R_A.
If R_C has not been defined, default is R_B.
If R_D has not been defined, default is R_A.

4. If TH_B is blank, default is TH_A.
If TH_C is blank, default is TH_B.
If TH_D is blank, default is TH_A.

5. If P_B is blank, default is P_A.
If P_C is blank, default is P_B.
If P_D is blank, default is P_A.

6. If DIV1 has not been defined, default is 1.
If DIV2 has not been defined, default is 1.
If DIV3 is blank, default is DIV1.
If DIV4 is blank, default is DIV2.

7. Parameters R_i, ϕ_i, DIVi, L, and θ which are passed from previously generated modules will override any values specified for this module.

8. This module may be used to generate cylindrical shapes by specifying $\phi_2 = \phi_4 = 90^\circ$. Annular plates may be generated by specifying $\phi_2 = \phi_4 = 0^\circ$ or 180°.

9. If ϕ_4 is blank, default is ϕ_2.

10. The following relationship must hold for the number of divisions on the edges of the module whenever DIV1 = DIV3:

$$|\text{DIV2} - \text{DIV4}| \leq \text{DIV1}.$$

Similarly, the following relationship must hold whenever DIV2 = DIV4:

$$|\text{DIV1} - \text{DIV3}| \leq \text{DIV2}.$$

11. The following relationship must hold for the number of divisions on the edges of the module whenever both DIV1 $\neq$ DIV3 and DIV2 $\neq$ DIV4:

$$|DIV1 - DIV3| < \min (DIV2, DIV4), \text{ and}$$

$$|DIV2 - DIV4| < \min (DIV1, DIV3).$$

12. With this module the user may specify $\phi_i \leq 0^\circ$ or $\phi_i \geq 180^\circ$; however, the above references to the directions to left and right must be reversed.

13. If M is zero or blank, a default number of 1 will be provided.

14. Modules which close a 360 degree ring (side 4 lies in the $\theta = 0^\circ = 360^\circ$ plane) must specify θ, even if it has been previously defined by another module.

Input Data Card **CONEND** Cone-Shaped End Module

Description: Defines a cone-shaped end module; data generated from apex to base (Figure 1.12).

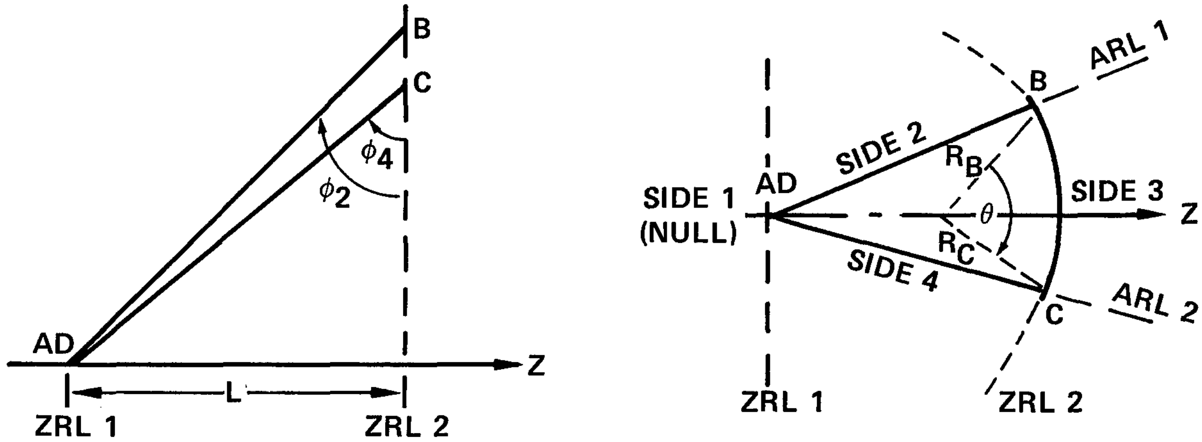


Figure 1.12 – Geometry of the CONEND Module

Format and example:

1	2	3	4	5	6	7	8	9	10
CONEND	ZRL1	ARL1	ZRL2	ARL2	ϕ_2	ϕ_4	L	θ	+IDENT1
CONEND	7	2	2	4	60.0		10.0	45.0	+ CE101

1	2	3	4	5	6	7	8	9	10
+IDENT1	R_B	TH_B	P_B	DIV3	M				+IDENT2
+CE101	2.0	1.0	1.0	6	1				+ CE102

1	2	3	4	5	6	7	8	9	10
+IDENT2	R_C	TH_C	P_C	DIV2	IEQ				+IDENT3
+CE102		1.5	1.0	8	1				+ CE103

1	2	3	4	5	6	7	8	9	10
+IDENT3		TH_{AD}	P_{AD}						
+CE103		1.5	1.5						

Field	Contents
ZRL1	Z-reference line number bounding the module at the apex, on the left (side 1, integer > 0)
ARL1	A-reference line number bounding the module at the top side (side 2, integer > 0)
ZRL2	Z-reference line number bounding the module on the right (side 3, integer > 0)
ARL2	A-reference line number bounding the module on the bottom (side 4, integer > 0)
ϕ_2	Angle between the negative r-direction and the vector BA along ARL1; clockwise is positive ($-90^\circ < \phi_2 < 90^\circ$, degrees)
ϕ_4	Angle between the negative r-direction and the vector CD along ARL2; clockwise is positive ($-90^\circ < \phi_4 < 90^\circ$, degrees)
L	Length of the projection of the module on the Z-axis (real ≥ 0)
θ	Angular width of the module (real, degrees)
R_B, R_C	Radii at corners B and C (real > 0)
TH_B, TH_C, TH_{AD}	Thicknesses at corners B, C, and the apex AD (real > 0)
P_B, P_C, P_{AD}	Pressures at corners B, C, and the apex AD (real)
DIV2, DIV3	Number of divisions along sides 2 and 3 (integer ≥ 1)
M	Material identification number (integer ≥ 0)
IEQ	Intersection equivalence flag (blank or 1)

Remarks:

1. The two Z-reference lines must have different numbers and may not be reference lines which have been equivalenced.

2. The two A-reference lines must have different numbers and may not be reference lines which have been equivalenced.

3. The number of divisions must be the same for sides 2 and 4.

4. DIV3 cannot be greater than DIV2.

5. This module may be used to model flat surfaces by setting $\phi_2 = \phi_4 = 0^o$.

6. If M is zero or blank, a default number of 1 will be provided.

7. R_A and R_D are automatically set to zero. R_B must be defined. If R_C has not been defined, default is R_B.

8. If TH_{AD} is blank, default is TH_B. If TH_C is blank, default is TH_B.

9. If P_{AD} is blank, default is P_B. If P_C is blank, default is P_B.

10. Parameters R_i, ϕ_i, L, DIVi and θ which are passed from previously generated modules will override any values specified for this module.

11. If $\phi_i < 0^o$, then the above references to the directions left and right must be reversed.

12. If ϕ_4 is blank, default is ϕ_2.

13. If IEQ is 1, no intersection equivalences will be invoked for the apex points. This flag must be set for one module, within a set of CONEND modules, which form a 360-degree cone. It should not be set for other applications.

14. Modules which close a 360-degree cone (side 4 lies in the $\theta = 0^o = 360^o$ plane) must specify θ, even if it has been previously defined by another module.

Input Data Card CONENDR Cone-Shaped End Module

Description: Defines a cone-shaped end module; data generated from base to apex (Figure 1.13).

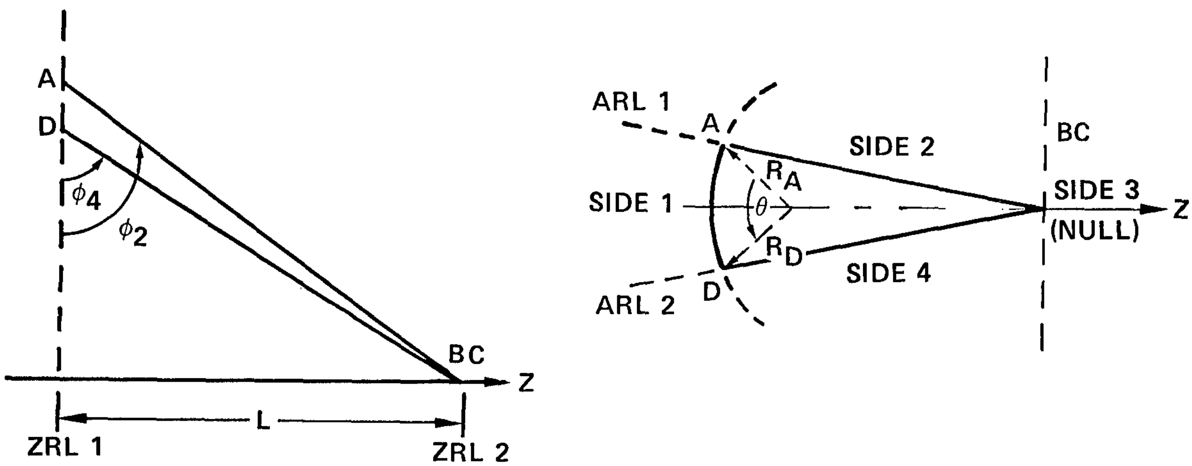


Figure 1.13 – Geometry of the CONENDR Module

Format and example:

1	2	3	4	5	6	7	8	9	10
CONENDR	ZRL1	ARL1	ZRL2	ARL2	ϕ_2	ϕ_4	L	9	+IDENT1
CONENDR	2	2	7	4	60.0			45.0	+ CR101

1	2	3	4	5	6	7	8	9	10
+IDENT1	R_A	TH_A	P_A	DIV1	M				+IDENT2
+CR101	1.5	1.0	1.0	6					+ CR102

1	2	3	4	5	6	7	8	9	10
+IDENT2	R_D	TH_D	P_D	DIV2	IEQ				+IDENT3
+CR102	1.5	1.0	1.0	8					+ CR103

1	2	3	4	5	6	7	8	9	10
+IDENT3		TH_{BC}	P_{BC}						
+CR103		1.5	1.5						

Field	Contents
ZRL1	Z-reference line number bounding the module on the left (side 1, integer > 0)
ARL1	A-reference line number bounding the module at the top (side 2, integer > 0)
ZRL2	Z-reference line number bounding the module at the apex, on the right (side 3, integer > 0)
ARL2	A-reference line number bounding the module on the bottom (side 4, integer > 0)
ϕ_2	Angle between the negative r-direction and the vector AB along ARL1; counterclockwise is positive ($-90^{\circ} < \phi_2 < 90^{\circ}$, degrees)
ϕ_4	Angle between the negative r-direction and the vector DC along ARL2; counterclockwise is positive ($-90^{\circ} < \phi_4 < 90^{\circ}$, degrees)
L	Length of the projection of the module on the Z-axis (real ≥ 0)
θ	Angular width of the module (real, degrees)
R_A, R_D	Radii at corners A and D (real > 0)
TH_A, TH_D, TH_{BC}	Thicknesses at corners A, D, and the apex BC (real > 0)
P_A, P_D, P_{BC}	Pressures at corners A, D, and the apex BC (real)
DIV1, DIV2	Number of divisions along sides 1 and 2 (integer ≥ 1)
M	Material identification number (integer ≥ 0)
IEQ	Intersection equivalence flag (blank or 1)

Remarks:

1. The two Z-reference lines must have different numbers and may not be reference lines which have been equivalenced.

2. The two A-reference lines must have different numbers and may not be reference lines which have been equivalenced.

3. The number of divisions must be the same for sides 2 and 4.

4. DIV1 cannot be greater than DIV2.

5. This module may be used to model flat surfaces by setting $\phi_2 = \phi_4 = 0^o$.

6. If M is zero or blank, a default number of 1 will be provided.

7. R_B and R_C are automatically set to zero. R_A must be defined. If R_D has not been defined, default is R_A.

8. If TH_{BC} is blank, default is TH_A. If TH_D is blank, default is TH_A.

9. If P_{BC} is blank, default is P_A. If P_D is blank, default is P_A.

10. Parameters R_i,ϕ_i,DIVi, L, and θ which are passed from previously generated modules will override any values specified for this module.

11. If $\phi_i < 0^o$, then the above references to the directions left and right must be reversed.

12. If ϕ_4 is blank, default is ϕ_2.

13. If IEQ is 1, no intersection equivalences will be invoked for the apex points. This flag must be set for <u>one</u> module, within a set of CONENDR modules, which form a 360-degree cone. It should not be set for other applications.

14. Modules which close a 360-degree cone (side 4 lies in the $\theta = 0^o = 360^o$ plane) must specify θ, even if it has been previously defined by another module.

Input Data Card GEQU General Data Equivalence

Description: Defines equivalences between any data groups stored in KEY-CHAIN storage.

Format and example:

1	2	3	4	5	6	7	8	9	10
GEQU	ZRLD	ARLD	COMPD	LEVELD	ZRLI	ARLI	COMPI	LEVELI	+IDENT
GEQU	12	16	2	1	13	16	1	1	

Field	Contents
ZRLD	Z-reference line used to define the dependent reference point (integer > 0)
ARLD	A-reference line used to define the dependent reference point (integer > 0)
COMPD	Data component at the dependent reference point which is to be equivalenced (integer 1, 2, or 3)
LEVELD	Data level, in KEY-CHAIN storage, of the data to be equivalenced (integer > 0)
ZRLI	Z-reference line used to define the independent reference point (integer > 0)
ARLI	A-reference line used to define the independent reference point (integer > 0)
COMPI	Data component at the independent data point (integer 1, 2, or 3)
LEVELI	Data level, in KEY-CHAIN storage, of the independent data component (integer > 0)

Remarks:

1. See Section 1.1.5 for a discussion of equivalencing and Section 2.2.1 for a discussion of the management of KEY-CHAIN storage.

2. Data component numbers are defined:

   1 - data along a Z-reference line

   2 - data along an A-reference line

   3 - data at an intersection point.

3. Notice that, once a data group is equivalenced, all higher levels of data at that point are also equivalenced.

4. Up to five continuation cards are permitted for one logical card (fields 2 through 9 have the same interpretation as on the first card). There is no advantage in using continuation cards as opposed to specifications on separate logical cards.

5. The user must insure that equivalence specifications are not circular. There are no other restrictions regarding which data groups may or may not be equivalenced.

Input Data Card IEQU Intersection Point Equivalence

Description: Defines equivalence between intersection points of A- and Z-reference lines.

Format and example:

1	2	3	4	5	6	7	8	9	10
IEQU	ZRLDI	ARLD1	ZRLI1	ARLI1	ZRLD2	ARLD2	ZRLI2	ARLI2	+IDENT1
IEQU	4	2	4	1	4	3	4	1	+ C01

1	2	3	4	5	6	7	8	9	10
+IDENT1	ZRLD3	ARLD3	ZRLI3	ARLI3	etc.				
IEQU	4	4	4	3					

Field	Contents
ZRLDi, ARLDi	Pair of Z- and A-reference line numbers defining a dependent intersection point--one that will be equivalenced to point ZRLIi, ARLIi (integer > 0).
ZRLIi, ARLIi	Pair of Z- and A-reference line numbers defining an independent intersection point--one that will be used for grid point numbering and global reference (integer > 0).

Remarks:

1. See Section 1.1.5 for a discussion of equivalencing.

2. The user must insure that equivalence specifications are not circular. There are no other restrictions regarding which reference line intersections may or may not be equivalenced.

Input Data Card ZEQU Z-Reference Line Equivalence

Description: Defines equivalence between Z-reference lines.

Format and example:

1	2	3	4	5	6	7	8	9	10
ZEQU	ZRLD1	ZRLI1	ZRLD2	ZRLI2	ZRLD3	ZRLI3	etc.		
ZEQU	5	3	1	2					

Field	Contents
ZRLDi	Dependent Z-reference line number that will be equivalenced to ZRLIi (integer > 0).
ZRLIi	Independent Z-reference line number that will be used for numbering grid points and is equivalenced to ZRLDi (integer > 0).

Remarks:

1. Maximum number of Z-reference lines is 39.
2. See Section 1.1.5 for a discussion of equivalencing.
3. The user must insure the equivalence specifications are not circular. There are no other restrictions regarding which reference lines may or may not be equivalenced.

2. PROGRAMMER'S INFORMATION

2.1 PROGRAM ORGANIZATION

2.1.1 General Structure

Using the data generator modules as a set of independent data generation programs requires that two passes be made through the user's data. The first pass scans the data to establish the mechanism necessary for communication between modules. Internal identification tags are assigned to the user's specifications, and storage areas are reserved for information to be passed at module boundaries. After the first pass, an interlude phase is entered which completes preliminary processing by propagating specifications to undefined areas and assigning default values to those quantities which are still undefined.

The second pass through the user's data, which is the actual data generation pass, has two phases. The first phase generates data for surface modules and the second phase generates all other data (primarily from frame, boundary condition, and non-uniform load modules).

When the data generation has been completed, various data files are merged onto one file which can be passed from the program as punched cards, or as a tape or disk file to be analyzed by the NASTRAN program. Computer plots of the model are then generated for user verification.

Table 2.1 lists the subroutines corresponding to major operational steps of the program and summarizes their primary functions. The process completion indicator (seventh word in the OPTION common block) is used by the subroutine ABORT to determine which storage areas will produce meaningful dumps. With the exception of subroutine INSORT this indicator is set after control has been returned from the subroutine listed.

TABLE 2.1 - PRIMARY PROCESSING SUBROUTINES

Subroutine Name	Process Completion Indicator	Function
BEGIN	0	
SETUP	10	Reads and interprets control cards, sets processing options and page heading.
GOOGAN	20	Transforms NASTRAN format BULK DATA to a FORTRAN readable format.
INSORT	30	First pass through data. Assigns internal numbers and boundary storage areas.
	40	Interlude processing. Completes storage assignments, processes reference line equivalences, assigns default values to mesh specifications.
CUTUP	50	Begin second pass processing. Generates data for surface modules.
FRAMIT	60	Complete second pass processing. Generates data for frame, boundary condition, and non-uniform pressure loading.
ASSMBL	70	Merges generated data onto one file to be plotted and passed on to subsequent programs.
NASPLT	(STOP)	Generates computer plots of idealization.

2.1.2 Program Conventions

A few general programming conventions were adopted at the outset of the development of the program. As work progressed, it became evident that adopting other conventions would simplify future modifications and additions. Some of the conventions have not been rigidly followed

but all current development follows these guides and existing code is being changed to conform.

2.1.2.1 Headings and Pagination of Printed Output. All printed output should be accounted for by the PRINT subroutine (Section 2.6.5). This will insure proper pagination and will automatically print headings and subheadings at the top of each page.

2.1.2.2 Error Messages. Each error is assigned a number by the programmer and listed in the error message table (Section 3). Non-fatal error messages and information messages are preceded by a blank line and have the form:

"*******bbXXX1ZZZbb message text ..."

where XXX is the current step number as indicated by the seventh word of the OPTION common block and ZZZ is the error message number.

Fatal error messages have two levels of severity. The most severe (level 3) fatal errors are those which require that the application be aborted without any further processing. It is the programmer's responsibility to print a fatal message and return to the main program with the value 3 stored in the NSR word of the OPTION common block. Less severe (level 2) fatal errors are those which continue processing for data checking purposes, but suppress NASTRAN execution. It is the programmer's responsibility to print a fatal error message and to store the value 2 in NSR word of OPTION. It is also the programmer's responsibility to count all such errors occurring in his module and to initiate a level 3 error termination if the number of errors exceeds the error limit (value stored in the fourteenth word of OPTION). Fatal error messages are preceded by a blank line and have the form

"*FATAL*bbXXXYZZZbb message text ..."

where XXX is the current step number as indicated by the seventh word of the OPTION common block, Y is the severity level (2 or 3), and ZZZ is the error message number.

2.1.2.3 Structural Module Specifications. In order to standardize input formats for the user and to simplify the global access to various quantities required for mesh propagation, the following conventions are proposed as standard basic specifications for structural modules:

<u>All Modules</u>

- The first data field on the card will contain the module name (always must be specified).

<u>Surface Modules</u>

- The second through the fifth data fields on the first card will contain the identification numbers of the reference lines defining the module (always must be specified).
- The first four continuation cards will contain information about the four (three) edges of the structural module.
- The fifth field on each of the first four continuation cards will contain the number of divisions which will be made along one boundary of the module in establishing a finite element mesh (optionally specified by automatic mesh propagation by the program, or by assigned defaults).
- Provision should be made for absolute specification of the coordinates of the intersections of the reference lines bounding the module (usually obtained by the program as propagated quantities from adjacent modules).

<u>Solid Modules</u>

- No recommendations at this time.

<u>Stiffener, Boundary Condition, and Loading Condition Modules</u>

- The second data field will contain the identification number of the reference line along which this stiffener is positioned (always must be specified).
- The third and fourth data fields will contain the reference line identification numbers defining the starting and ending points of the stiffener (a stiffener will be generated along the entire length of the reference line, as defined by surface or solid module specifications, if no starting and ending points are specified).

2.1.2.4 Subroutine Conventions. The only conventions established for the writing of subroutines are:

- FORTRAN non-standard returns should not be used;
- all references to files should be symbolic, with the file variables appearing in the subroutine calling arguments.

2.1.2.5 Communication between Modules. The program uses data files as its primary method of communication. The accession of data to be processed by each module and the transfer of output generated by each module are accomplished using files. Necessary parameters and flags are passed in the OPTION common block (Section 2.1.3). Global geometric information is passed in the KEYCHN common block and is accessed using the LOCKS and LOCKIT subroutines.

2.1.2.6 Standard Library Subroutines. A number of standard library routines (SIN, ATAN, etc.) are used throughout the program. Since these subroutines should be available as standard software components on any computer that would accommodate the program, they have not been listed in the "Subroutines Called" sections of the subroutine descriptions.

2.1.3 Global Common Storage Areas

The program has two common areas which apply to all segments of the program. One area called OPTION contains parameters which control the execution of the various program parts. Table 2.2 describes these parameters. The second area called SET is used only to pass page heading and output titling information to the subprogram PRINT and is described with that program in Section 2.6.5.

The OPTION area is initialized in subroutine SETUP from user control card specifications (Section 2.3.1). An additional execution control card

$$\text{OPTION(J)} = \text{K}$$

permits the user to store the value K in the J^{th} word of the OPTION common area. A second execution control card,

TABLE 2.2 - OPTION COMMON AREAS

Word Index	Acceptable Values	Function
1	1 0	Connection cards to be generated No connection cards to be generated
2	1 0	PLOAD cards to be generated No PLOAD cards to be generated
3	1 0	FORCE cards to be generated No FORCE cards to be generated
4	1 0	SPC cards to be generated No SPC cards to be generated
5	1 0	MAT1 cards to be generated if missing No MAT1 cards to be generated if missing
6	0 1 -1	Program decides which errors are fatal All errors considered fatal No errors considered fatal
7	≥0	Indicates the current phase of data generation processing (See Section 2.1.1 for description)
8	1 0	Generated data to be punched on cards Generated data not to be punched on cards
9	1	NASTRAN run to be executed following data generation run. Additional data cases will be ignored.
	0	No NASTRAN run to follow. Multiple data cases will be processed.
10	Not Used	
11	≥0	Number of shell modules to be processed
12	≥0	Number of frame modules to be processed
13	≥0	Number of boundary condition modules to be processed
14	≥0	Maximum number of level 2 errors permitted per processing step
	50	Default
15	>10	Number of lines per page (including title and heading lines)
	55	Default

TABLE 2.2 - (continued)

Word Index	Acceptable Values	Function
16	≥0	Number of generated connection cards
17	≥0	Number of generated GRID cards
18	≥0	Number of loading cards generated
19	0	No data storage areas to be dumped by subroutine ABORT
	1	Data storage areas dumped in edited format by subroutine ABORT
	-1	Data storage areas dumped in unedited format by subroutine ABORT
	2	Data storage areas dumped in edited format by subroutine ABORT following a fatal (level 3) error. Default
20	>0	Number of words of a data storage area to be dumped by subroutine ABORT when an unedited dump is requested.
	2000	Default
21	1	Maximum printing of error and information messages
	0	Minimum printing of messages
22	>0	NASTRAN rigid format number for data being generated
23	>0	Maximum number of rigid formats available
	12	Default
24	1	Structural plots to be generated for this data case
	0	No structural plots to be generated for this data case
25	≥25	Number of words in working OPTION common area
	50	Default
OPTION(25)		
+1 (NSR)	1	No errors detected at this point
	2	Level 2 errors (conditionally fatal) detected at this point
	3	Level 3 errors (fatal) detected at this point

$$\text{SETOPT} = \begin{Bmatrix} \text{HIGH} \\ \text{LOW} \end{Bmatrix}$$

permits the user to issue blanket option requests for debugging assistance. Table 2.3 indicates OPTION common area changes produced by the SETOPT card (the OPTION common area will be dumped by all calls to subroutine ABORT, regardless of the control card specifications in effect).

TABLE 2.3 - OPTION COMMON CHANGES PRODUCED BY THE SETOPT CARD

SETOPT = HIGH

OPTION WORD	Value
6	-1
14	100
19	1
20	500
21	1

SETOPT = LOW

OPTION WORD	Value
1	0
2	0
3	0
4	0
5	0
14	1
19	0
21	0

2.1.4 Program Overlay Structure

The data generator program operates on the CDC 6700 and CDC 6400 computers at NSRDC. The program has been compiled using the FORTRAN Extended (FTN) compiler and executes from linked overlays using the NASTRAN LINKEDITOR/LOADER. The program is divided into two main segments: one for generating data and one for plotting the generated model. Figures 2.1, 2.2, and 2.3 show the overlay control cards for the main driving link and the two subordinate links, respectively.

```
LINKEDIT OUTFILE=DATGEN(S)  PARAM(6)15000
LIBRARY LIBA

LINK 0 *PROGRAM DRIVER
INCLUDE LIBA(SCRIBE)
INCLUDE LIBA(PRINT)
INCLUDE LIBA(PLOTDD)
INCLUDE LIBA(FLAGSV)
INCLUDE LIBA(SHFT1V)
INCLUDE LIBA(SHFT2V)
INCLUDE LIBA(ORAV)
ENTRY SCRIBE
END
```

Figure 2.1 - Driver Link Overlay Control Cards

```
LINK1      *DATA GENERATION LINK
INCLUDE    LIBA(DATAGEN)
INCLUDE    LIBA(BLKDATA(TYPE))
INCLUDE    LIBA(SWITCH)
INCLUDE    LIBA(ERRMSG)
INCLUDE    LIBA(ABORT)
INCLUDE    LIBA(LOCKS)
INCLUDE    LIBA(LOCKIT)
INCLUDE    LIBA(POOLZZ)
INCLUDE    LIBA(POOLIT)
INCLUDE    LIBA(POOLPR)
INCLUDE    LIBA(POOLDT)
INCLUDE    LIBA(POOLPS)
INCLUDE    LIBA(POOLDS)
INCLUDE    LIBA(FETCH)
INCLUDE    LIBA(FETCH1)
INCLUDE    LIBA(STOW)
INCLUDE    LIBA(STOW1)
INCLUDE    LIBA(PURGE)
INCLUDE    LIBA(DMPOOL)
OVERLAY    A
INCLUDE    LIBA(SETUP)
INCLUDE    LIBA(MASQ)
INCLUDE    LIBA(XRCARD)
INCLUDE    LIBA(GOOGAN)
OVERLAY    A
INCLUDE    LIBA(INSORT)
INCLUDE    LIBA(XTRACT)
INCLUDE    LIBS(SPACE)
OVERLAY    A
INCLUDE    LIBA(ASSMBL)
OVERLAY    A
INCLUDE    LIBA(CUTUP)
INCLUDE    LIBA(READS)
INCLUDE    LIBA(REF)
INCLUDE    LIBA(TERP)
INCLUDE    LIBA(CONE)
INCLUDE    LIBA(QUADS)
INCLUDE    LIBA(PROPER)
INCLUDE    LIBA(CONEND)
INCLUDE    LIBA(TRI)
INCLUDE    LIBA(CEPROP)
ENTRY  DATGEN
END
```

Figure 2.2 - Data Generation Link Overlay Control Cards

```
LINK2      *STRUCTURE PLOTTING LINK
INCLUDE    LIBA(NASPLT)
OVERLAY    AA
INCLUDE    LIBA(COORD)
INCLUDE    LIBA(CORD12)
INCLUDE    LIBA(FINDC)
INCLUDE    LIBA(FINDG)
INCLUDE    LIBA(SYS)
OVERLAY    AA
INCLUDE    LIBA(PLOT)
INCLUDE    LIBA(GLABEL)
INCLUDE    LIBA(XFRAME)
INCLUDE    LIBA(MODEL)
INCLUDE    LIBA(CENTRE)
INCLUDE    LIBA(IDFRMV)
INCLUDE    LIBA(APRNTV)
INCLUDE    LIBA(BNBCDV)
INCLUDE    LIBA(CAMRAV)
INCLUDE    LIBA(CHSIZV)
INCLUDE    LIBA(CNTCDC)
INCLUDE    LIBA(CNTIBM)
INCLUDE    LIBA(CTL4V)
INCLUDE    LIBA(DOTLNV)
INCLUDE    LIBA(ERMRKV)
INCLUDE    LIBA(ERRLNV)
INCLUDE    LIBA(ERRNLV)
INCLUDE    LIBA(FORMV)
INCLUDE    LIBA(FRAMEV)
INCLUDE    LIBA(GRID1V)
INCLUDE    LIBA(HOLDIV)
INCLUDE    LIBA(HOLLV)
INCLUDE    LIBA(ID4G)
INCLUDE    LIBA(INTBCD)
INCLUDE    LIBA(LABLV)
INCLUDE    LIBA(LINEV)
INCLUDE    LIBA(LINRV)
INCLUDE    LIBA(NONLNV)
INCLUDE    LIBA(NXNYV)
INCLUDE    LIBA(NXV)
INCLUDE    LIBA(PAGE40)
INCLUDE    LIBA(PLOTV)
INCLUDE    LIBA(POINTV)
INCLUDE    LIBA(PRINTV)
INCLUDE    LIBA(RITE2V)
INCLUDE    LIBA(RITSTV)
INCLUDE    LIBA(SCERRV)
INCLUDE    LIBA(SCLSAV)
INCLUDE    LIBA(SETCIV)
INCLUDE    LIBA(SETMIV)
INCLUDE    LIBA(TABLES)
INCLUDE    LIBA(TABL1V)
INCLUDE    LIBA(TABL2V)
INCLUDE    LIBA(TABL3V)
INCLUDE    LIBA(VCHARV)
INCLUDE    LIBA(VECTRV)
INCLUDE    LIBA(XAXISV)
INCLUDE    LIBA(XMODV)
INCLUDE    LIBA(XSCALV)
INCLUDE    LIBA(INCRV)
ENTRY NASPLT
END
```

Figure 2.3 - Structure Plotting Link Overlay Control Cards

2.1.5 Program Execution

On the CDC computer the program can be executed using standard procedures or the program can be executed from a LINKEDITOR random overlay file. The acceptable forms of control cards for program execution are listed in Table 2.4.

TABLE 2.4 - SYSTEM CONTROL CARDS FOR PROGRAM EXECUTION

1. progname.
2. progname(input, output, punch, genout, pltout)
3. progname. ATTACH
4. progname(input, output, punch, genout, pltout)ATTACH

Forms 1 and 2 in the table are used when the program is stored on a standard SCOPE operating system (sequential) file and forms 3 and 4 apply to LINKEDITOR random files. Forms 1 and 3 cause default file names to be used for the interface with the program while forms 2 and 4 allow the user to override the default names. A description of the program's external files is given in Table 2.5.

TABLE 2.5 - DATA GENERATOR EXTERNAL FILES

Symbol (Table 2.4)	Default Name	Contents
progname	No default	File containing program
input	INPUT	Data cards read by program
output	OUTPUT	Printed listing from program
punch	PUNCH	Punched cards generated by program (a copy of GENOUT)
genout	GENOUT	Generated NASTRAN card deck
pltout	PLTOUT	Plot file for SC 4020 plotter (tape file only, written at 556 BPI density in S format)

The examples given below illustrate some typical applications when the program is stored on the file CUTUP.

(1)
```
LABEL, PLTOUT, L=MYPLOTTAPE, D=HI, F=S, R.
REQUEST, GENOUT, *PF.
CUTUP.
CATALOG, GENOUT, MYFILENAME, ID=1234567890.
```

(2)
```
LABEL, TAPE, L=MYTAPE, R, D=HY.
CUTUP(, , TAPE)ATTACH
NASTRAN(GENOUT)ATTACH
```

(3)
```
CUTUP(, NULL)
NASTRAN(GENOUT)ATTACH
```

In (1) the program is stored on a sequential file and the generated data cards are to be stored on a permanent file called "MYFILENAME." Plot information will be generated on the tape PLTOUT for SC 4020 plotting. In (2) the program is stored in random format, a copy of the generated data (the normal punch copy) is to be stored on tape, and the generation run will be followed by a NASTRAN run. In (3) the program is stored on a sequential file, the printed listing is to be deleted, and a NASTRAN run will follow.

2.1.6 Execution Monitor Program - DATGEN

Function: To control the sequence of operations during a data generation program run.

Common Blocks:

OPTION — See Section 2.1.3

KEYCHN — See Section 2.2.2

	Word	Type	Contents
TYPE	1	integer	Number of words in processing control alphabet - 40
	2-41	alphanumeric	Processing control alphabet: 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, *, +, blank, $, A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

Word	Type	Contents
42	integer	Number of words in Bulk Read Dictionary - 11
43-53	alphanumeric	Bulk Read Dictionary: MAT1, blank, MAT2, blank, MAT3, blank, BEGI, NbBu, LKbb, ENDD, ATAb
54	integer	Number of words in Bulk Write Dictionary - 22
55-76	alphanumeric	Bulk Write Dictionary: CBAR, blank, PBAR, blank, CQUA, D2bb, PQUA, D2bb, CTR1, A2bb, PTR1, A2bb, PLOA, Dbbb, GRID, blank, SPCb, blank, CORD,2Cbb, PLOT, ELbb
77	integer	Number of words in "S" Dictionary - 6
78-83	alphanumeric	"S" Dictionary: CONE, blank, CONE, NDbb, CONE, NDRb
84-87	alphanumeric	Four words of "$ $ $ $"
88	integer	Number of words in "F" Dictionary - 4
89-92	alphanumeric	"F" Dictionary: FRAM, Tbbb, RING, Tbbb
93-98	alphanumeric	Six words of "$ $ $ $"
99	integer	Number of words in "B" Dictionary - 2
100-109	alphanumeric	Ten words of "$ $ $ $"
110	integer	Number of words in "O" Dictionary - 2
111-112	alphanumeric	"O" Dictionary: $END, blank
113-120	alphanumeric	Eight words of "$ $ $ $"
121	integer	Number of words in "Q" Dictionary - 6
122-127	alphanumeric	"Q" Dictionary: ZEQU, blank, AEQU, blank, IEQU, blank

Entry Point: DATGEN

Calling Sequence: Main Program

Subroutines Called: ABORT, ASSMBL, CUTUP, FRAMIT, GOOGAN, INSORT, NASPLT, PLASSM, POOLIT, PRINT, SETUP

Files Defined:

File Number	File Name	Function
1	TAPE1	Scratch, transfer of data to NASPLT
5	INPUT	User data to program
6	OUTPUT	Printed listing and message file
7	PUNCH	Punched card file
8	TAPE8	Scratch
9	TAPE9	Scratch
10	TAPE10	Scratch
11	TAPE11	Scratch
12	TAPE12	Scratch
13	TAPE13	Scratch
14	TAPE14	Scratch
15	TAPE15	Scratch
16	GENOUT	Generated card image file
48	PLTOUT	Structural plot file

Program Messages:

Number	Level	Text
40	1	END OF APPLICATION
41	3	EXECUTION OF NASTRAN SUPPRESSED DUE TO ABOVE ERRORS

Method: DATGEN is a driver program which calls the various functional subroutines which make up the data generator. The value of the NSR word of the OPTION common block is checked to determine whether processing should continue after each functional step. The NSR word is also checked at the completion of processing for each data case to determine whether it is permissible to begin a structural analysis. Multiple data cases will be processed unless the ninth word of the OPTION

common block indicates that an analysis step follows this data case.

Remarks:

1. When the program was subdivided into several overlay levels, a dummy driver program (SCRIBE) was added to call the DATGEN and NASPLT overlays; however, DATGEN retains the sequence control as described.

2.2 DATA STORAGE

2.2.1 Storage Policy

In the development of the program an attempt has been made to store all generated data on external files as it is produced and to retain in core storage only the key information required for communication between modules. Other efforts have been made to conserve core storage and to manage variable length tables used by the program. Two core data management systems have been developed: the KEY-CHAIN data management technique for intermodule communication, and the POOLIT technique for managing tables and lists. Although POOLIT is an in-core storage scheme, it has been coded so that a paged table technique (using random disc storage) could be substituted to further reduce storage requirements without extensive program changes.

2.2.2 KEY-CHAIN Data Storage Method

KEY-CHAIN data storage is used to pass information among modules within a structure. It is also an integral component of the automatic mesh propagation feature of the program. This data storage method is constructed around a tree data structure with a directory to the data which is indexed by pairs of identification numbers of intersecting reference lines.

During the first pass through the user's data a basic amount of storage in the CHAIN array is allocated to store the coordinates of each grid point along the edges). The corner grid points are assigned space

independently to resolve the problems which arise because they belong to two edges. Edges which are common between two or more modules receive only one allocation. At the first reading of a user's data record the number of grid points along the edge of a module will not be defined if the user is expecting the mesh density to be propagated by the program. In this event a zero length allocation is made in the CHAIN array (pointer only) and the final assignment will be made during interlude processing after the first pass through the data.

In the course of data generation additional space may be required by modules which pass other information in addition to the coordinates of boundary grid points. The original allocation is referred to as the first level of storage and each subsequent allocation receives a level number which is one greater than the previous allocation. After the allocation is made, all information is stored and retrieved using this level number.

The index is a rectangular array (KEY) of three word blocks. The (i, j)th block in this array refers to data associated with the intersection of the i^{th} z-reference line and the j^{th} A-reference line. The first word in each block points to the level 1 storage region in the CHAIN array for the edge along a Z-reference line, the second to a storage region for an edge along an A-reference line, and the third to a storage region for the intersection of two reference lines. Figures 2.4 and 2.5 illustrate this method. Notice that there are multiple levels of data both along ZRL No. 1 and at the intersection and that allocation has been deferred along ARL No. 1.

Subroutines available to assist in the management of data stored using this method include XINIT, SPACE, LOCKIT, LOCKS, and DMKYCH. XINIT is an entry point in subroutine XTRACT which initializes KEY-CHAIN storage. Subroutine SPACE performs level 1 assignments of CHAIN storage space. Subroutine LOCKIT is used to

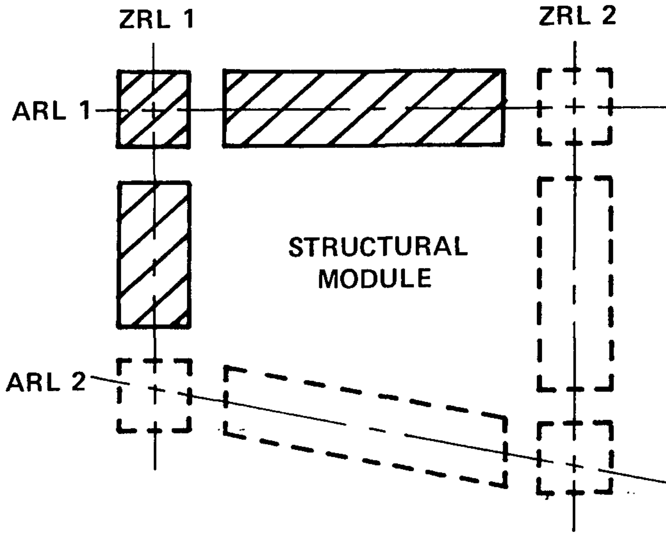


Figure 2.4 – Boundary Data Storage Associated with the Intersection of Two Reference Lines

locate a data block in the CHAIN array by referencing a pair of reference line indices and level numbers. Subroutine LOCKS is a specialized version of LOCKIT for referencing level 1 data only. Dumps of the KEY-CHAIN storage area can be produced by calling DMKYCH entry point in subroutine DMPOOL. All KEY-CHAIN arrays and parameters are located in the KEYCHN common block.

2.2.3 KEY-CHAIN Utility Program Specifications

Common Block:

COMMON/KEYCHN/KEY(3, 40, 19), CHAIN(3, 1500), LCHAIN, KCHAIN, KZ, KA, KZMAX, KAMAX, IEXTZ(2, 40), IEXTA(2, 19), KQZ, KQA, IEQZ(2, 40), IEQA(2, 19), THETA(19)

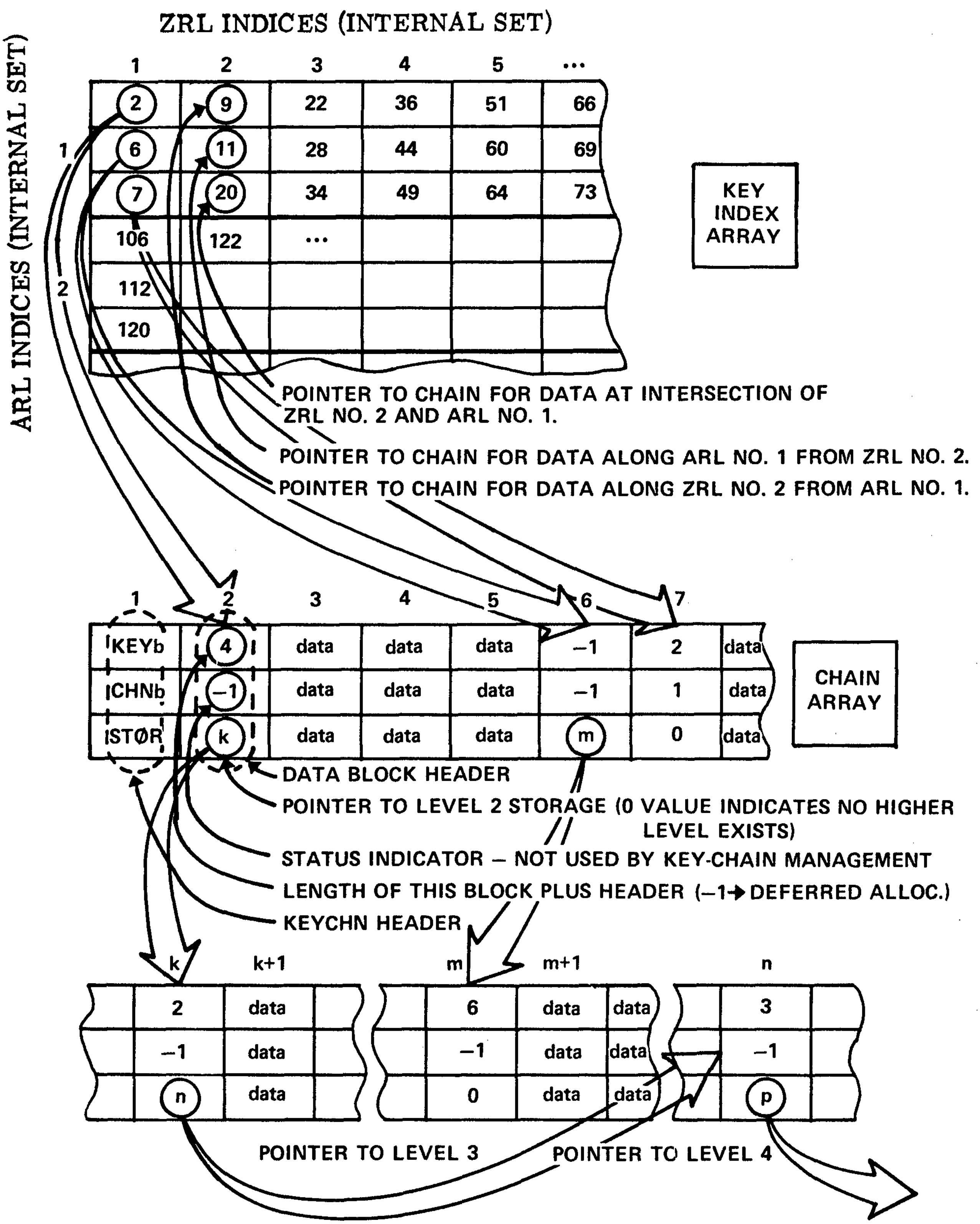


Figure 2.5 – KEY-CHAIN Data Storage Organization

Subroutines:

Name	Function
DMKYCH	Dumps KEY-CHAIN storage areas
LOCKIT	Basic routine which assigns KEY-CHAIN storage space and locates stored data
LOCKS	Specialized form of LOCKIT for level 1 data
SPACE	Assigns level 1 storage space and stores pointers for mesh propagation with zero-length requests
XINIT	Initializes KEY-CHAIN storage areas

Tables and Storage Areas:

CHAIN	Chained storage area
KEY	First level index to chained storage area
IEXTZ	Internal-external correspondence table for Z-reference lines
IEXTA	Internal-external correspondence table for A-reference lines
IEQZ	Equivalence correspondence table for Z-reference lines
IEQA	Equivalence correspondence table for A-reference lines
THETA	Not used
LCHAIN	Maximum number of three-word blocks available in the CHAIN area
KCHAIN	Current number of three-word blocks being used in the CHAIN area
KZMAX	Maximum number of Z-reference lines permitted per application
KAMAX	Maximum number of A-reference lines permitted per application
KZ	Current number of Z-reference lines defined in this application
KA	Current number of A-reference lines defined in this application
KQZ	Number of entries in IEQZ
KQA	Number of entries in IEQA

Miscellaneous Information:

Format of Data Lists in Chain Storage

Block Number	word 1	word 2	word 3
n	IDIV	ICOORD	LEVPTR
n+1	DATA1	DATA2	DATA3
⋮	⋮	⋮	⋮
n+IDIV-1	DATA1	DATA2	DATA3

Symbol	Definition
n	Beginning block number of this list
IDIV	Number of blocks in this data list. IDIV=-1 during first pass if no storage has been allocated for this list (because its length is not known). If IDIV $<$ -1 after first pass, the absolute value of IDIV points to the block number of a new area where this list now resides; however, the KEY index should reflect such a change and this pointer should not be required once all storage has been resolved. (For level 1 lists this will be the number of divisions along the segment of the reference line for which data is being stored in this list.)
ICOORD	Status indicator for data in this list. (If ICOORD=-1, the list is empty. For level 1 lists this will be the coordinate system identification for the grid point coordinates stored in this list.)
LEVPTR	Pointer containing the block number of the data list at the next level in this chain. LEVPTR=0 indicates that this is the last data list in the chain.
DATAi	Data item (for level 1 lists this is the i-th coordinate locating the grid point being stored).

Format of KEY Index:

See Figure 2.5.

Subroutine <u>DMKYCH</u> (Entry point in Subroutine DMPOOL)

Function: Dumps core storage areas associated with the KEY-CHAIN data management method.

Common Blocks: OPTION, SET, KEYCHN

Entry Point: <u>DMKYCH</u>

Calling Sequence: CALL DMKYCH(IFORM)

IFORM Dump format flag.

If IFORM > 0, dump will be restricted to regions in use as defined by the KEYCHN parameters. CHAIN array will be dumped with one 3-word block per printed line in the order defined by the KEY index. Thus KEY-CHAIN storage may be viewed as a three-dimensional array of variable length data groups. In the dump a data group which is n blocks long is printed as n lines, each preceded by the triple (component, ZRL, ARL). Only level 1 storage is dumped with this format.

If IFORM = 0, full sequential dumps of all parameters and arrays associated with KEY-CHAIN data storage will be printed.

If IFORM < 0, full sequential dumps of all parameters and arrays associated with KEY-CHAIN data storage will be printed except that the CHAIN array will be limited to -IFORM printed lines.

Subroutine Called: PAGE

Error Messages: None

Remarks:

1. With IFORM ≠ 0 data group dumps are limited to 100 words per block to guard against runaway printing in the event that block lengths have been destroyed.

Subroutine LOCKIT

Function: Performs general data management tasks for data in KEY-CHAIN storage. Assigns CHAIN storage space for all levels of data. Locates data referenced by reference line coordinates component number and level number.

Common Blocks: OPTION, KEYCHN, POOLTB

Entry Point: LOCKIT (Applicable for all levels of data.)

Calling Sequence: CALL LOCKIT(IZRL,IARL,IZA,LEVEL,IDIV,
LOC,IEZ,IEA)

IZRL, IARL	Reference line coordinates of requested data group (internal numbers, integer > 0)
IZA	Data component indicator; integer; if 1, data along ZRL if 2, data along ARL if 3, data at intersection of ZRL and ARL
LEVEL	Data group level number (integer > 0)
IDIV	The number of 3-word blocks in this data group including one header block (integer). If -1, only header will be (has been) allocated. A call with IDIV=0 indicates a data location request.
LOC	Pointer to first word of header for data group defined by IZRL, IARL, IZA, and LEVEL (integer ≥ 0). A return with LOC=0 indicates that no space has been assigned for the data group.
IEZ, IEA	Identification numbers to which IZRL and IARL have been equivalenced. These values will be set only after interlude processing is complete (when the seventh word of OPTION is ≥ 40).

Entry Point: LOCKS (Applicable to level 1 data only.)

Calling Sequence: CALL LOCKS(IZRL,IARL,IZA,IDIV,LOC,IEZ,IEA)

Arguments have same definition as for LOCKIT.

Subroutines Called: ABORT,FETCH,POOLPS,PRINT

Error Messages:

1. A level 3 error message number 50 will be issued and the program will terminate if CHAIN storage is exceeded. Either the length of the CHAIN array must be increased or the problem size reduced. An estimate of the total CHAIN space required for the job can be made from the reference line coordinates of the requested data group.

Remarks:

1. On CDC 6700 entry point LOCKS is a separate subroutine which calls LOCKIT with LEVEL=1.

2. POOL storage is used to store equivalences for intersection point data. This data group is located by the pointer in NDIREC(3) and is required throughout the processing of one job.

Subroutine SPACE

Function: To manage the assignment of level 1 data storage in the CHAIN array.

Common Blocks: OPTION, KEYCHN

Entry Point: SPACE

Calling Sequence: CALL SPACE(IZRL, IARL, IDIV, IZA, MZRL, MARL)

IZRL, IARL	Reference line coordinates of requested boundary region (internal numbers)
IDIV	Number of 3-word blocks requested, including one header block. If less than 1, indicates that assignment is to be deferred until interlude processing phase. In this case one 3-word header block is assigned and a length of -1 is entered in header block.
IZA	Component indicator; if 1, boundary along ZRL if 2, boundary along ARL if 3, intersection point
MZRL, MARL	Reference line coordinates of the boundary of the module which is opposite the requested region. Used only if nonzero and IDIV=-1. These coordinates facilitate mesh propagation during interlude processing. Applicable only to components 1 and 2.

Subroutines Called: ABORT, LOCKS

Error Messages:

1. A level 2 error message number 51 will be issued if, in processing a request for space, the length of the requested data group does not match the length in an earlier request.

Remarks:

1. If more than one request for space is made for a particular data group and the specifications are consistent, a return is made to the calling program with no action taken.

Subroutine XINIT (Entry point in Subroutine XTRACT)

Function: To initialize the KEY-CHAIN storage area

Common Blocks: KEYCHN

Entry Point: XINIT

Calling Sequence: CALL XINIT

Remarks:

See Section 2.2.3 for a description of the KEYCHN common block.

2.2.4 POOLED Data Storage Method

POOLED data storage is used to store tables and lists whose lengths vary widely with different program applications. This method is particularly convenient for tables which exist for just one phase of the data generation processing since the space which they occupy can be purged and later reassigned.

The POOL storage area is divided into constant length blocks, called records. A list is kept of the records which are currently empty in the pool and available for use. The first two words of each pool record are pointers used in the management of the POOL. The first pointer indicates the status of the record; either unused, in use as the first record in a data group, or in use indicating the address of the last previous record in the group. The second word points either to the next available word in the record, if the record is partially filled with data, or to the next succeeding record, if this record is full.

Utility routines are provided as entry points to subroutine POOLIT which permit the user to store data items, to store data pairs, to search lists of data items and data pairs, to retrieve items serially, to replace items in a list, and to purge all data groups or individual data groups. Dumps of the POOL storage area can be obtained in either an unsorted format or sorted serially by data group using subroutine DMPOOL.

2.2.5 POOLED Utility Program Specifications

Common Blocks: COMMON/POOLTB/IPDEX(50),IPOOL(1100), IDEEP,ILENP,IMAX,IPOINT,NDIREC(10),LDIREC

IPDEX	List of available records in the pool
IPOOL	Data storage array
IDEEP	Maximum number of records in the pool
ILENP	Length of a pool record (including two preceeding pointers)

IMAX	Length of the IPOOL array
IPOINT	Pointer to next available record in IPDEX
NDIREC	Directory array of pooled data groups
LDIREC	Length of NDIREC array

COMMON/OPTION/ Refer to Section 2.1.3

Subroutines:

Name	Function
DMPOOL	Dumps POOLED storage areas
FETCH	Retrieves data stored in POOLED storage
FETCH1	Streamlined version of FETCH
PLDATA	Sequentially stores one data word in POOLED storage
PLPAIR	Sequentially stores a pair of data words in POOLED storage
POOLDT	Specialized version of PLDATA
POOLDS	Searches for a data word in POOLED storage
POOLIT	Initializes POOLED data storage parameters
POOLPR	Specialized version of PLPAIR
POOLPS	Searches for a pair of data words in POOLED storage
PURGE	Purges a data group from POOLED storage
STOW	Replaces a word stored in POOLED storage
STOW1	Streamlined version of STOW

Subroutine DMPOOL

Function: Dumps core storage areas associated with the POOLED data management method.

Common Blocks: OPTION, SET, POOLTB

Entry Point: DMPOOL

Calling Sequence: CALL DMPOOL(IFORM)

IFORM Dump format flag.

If IFORM > 0, edited dumps will be produced listing each data group as defined in the NDIREC array in sequential order. All auxiliary parameters and arrays will be dumped in full with identifying annotation.

If IFORM = 0, full sequential dumps of all parameters and arrays associated with POOLED data storage will be printed.

If IFORM < 0, full sequential dumps of all parameters and arrays associated with POOLED data storage will be printed except that the pool array will be limited to -IFORM printed lines.

Entry Point: DMKYCH (Described in Section 2.2.3.)

Subroutines Called: FETCH, PAGE

Error Messages: None.

Subroutine FETCH (Entry point to Subroutine POOLIT)

Function: Retrieves data words stored in POOLED storage.

Common Blocks: POOLTB, OPTION

Entry Point: FETCH

Calling Sequence: CALL FETCH(IA, IDENT, LOCSET, LOCP)

IA	Data word retrieved by FETCH
IDENT	Pointer to the particular data group to be referenced; set by first call to PLDATA, PLPAIR, POOLDT, or POOLPR (if IDENT $\leq$ 0, program will return with no processing)
LOCSET	Pointer, within the data group, to the data word to be retrieved
LOCP	POOL index of the data word retrieved (if no data is stored for the requested data item, LOCP is set to zero)

Entry Point: FETCH1

Calling Sequence: CALL FETCH1(IA, LOCP)

IA	Data word to be retrieved by FETCH1
LOCP	POOL index of the data word to be retrieved

Subroutines Called: None.

Error Message:

Number 105 The word IDENT does not point to a legal data group in POOL storage.

Remarks:

1. Entry point FETCH is called to retrieve data by its sequential position in the data group (as specified by LOCSET).

2. Entry point FETCH1 is called to retrieve data using the POOL index of the data word (as specified by LOCP). The index value must have been obtained from a previous call to FETCH, STOW, PLDATA, PLPAIR, POOLDS, or POOLPS. This method of data retrieval is faster than using the FETCH code and should be used when applicable.

Subroutine PLDATA (Entry point in Subroutine POOLIT)

Function: Stores data sequentially in POOLED storage, one word per call.

Common Blocks: OPTION, POOLTB

Entry Point: PLDATA

Calling Sequence: CALL PLDATA(IA,IDENT,LOCSET,IPLOC)

IA	Data word to be stored
IDENT	Pointer to a particular data group; set by program on first call; referenced by program on all subsequent POOLED calls
LOCSET	Data group index of word stored
IPLOC	Storage mode indicator; a zero value indicates a normal mode request (the last PLDATA, PLPAIR, POOLDT, or POOLPR operation was not necessarily a call to store data in this group; IPLOC will be set to the location of the next available row in POOL storage by the program for subsequent accelerated mode requests); a non-zero value indicates an accelerated mode request (the last PLDATA, PLPAIR, POOLDT, or POOLPR operation stored data in this group and thus IPLOC has been set to the location of the next available row in POOL storage)

Entry Point: POOLDT

Calling Sequence: CALL POOLDT(IA,IDENT,IPLOC)

Subroutine Called: ABORT

Error Messages:

Number	100	POOL storage space exceeded
Number	108	Illegal IDENT in call

Remarks:

1. The first call to this subroutine will initialize the POOLED storage area if not already initialized.

2. Unless the word IDENT is located in the array NDIREC of common block POOLTB, the data group will not be dumped by calls to DMPOOL with IFORM $>$ 0.

3. Entry point POOLDT is used whenever the data group index is not required as the data is being stored.

Subroutine <u>PLPAIR</u> (Entry point in Subroutine POOLIT)

Function: Stores pairs of data words sequentially in POOLED storage, one pair per call.

Common Blocks: OPTION, POOLTB

Entry Point: <u>PLPAIR</u>

Calling Sequence: CALL PLPAIR(IA,IB,IDENT,LOCSET,IPLOC)

IA,IB	Pair of words to be stored
IDENT	Pointer to a particular data group; set by program on first call; referenced by program on all subsequent POOLED calls.
LOCSET	Data group index of the first word of the pair stored
IPLOC	Storage mode indicator; a zero value indicates a normal mode request (the last PLDATA, PLPAIR, POOLDT, or POOLPR operation was not necessarily a call to store data in this group; IPLOC will be set to the location of the next available row in POOL storage by the program for subsequent accelerated mode requests); a non-zero value indicates an accelerated mode request (the last PLDATA, PLPAIR, POOLDT, or POOLPR operation stored data in this group and thus IPLOC has been set to the location of the next available row in POOL storage)

Entry Point: <u>POOLPR</u>

Calling Sequence: CALL POOLPR(IA,IB,IDENT,IPLOC)

Subroutine Called: ABORT

Error Messages:

Number	100	POOL storage space exceeded
Number	108	Illegal IDENT in call

Remarks:

1. The first call to this subroutine will initialize the POOLED storage area if not already initialized.

2. Data stored in a data group using PLPAIR or POOLPR should not be intermixed with data stored using PLDATA or POOLDT unless precautions are taken to insure that there are an even number of items in the group before using POOLPS to retrieve data.

3. Unless the word IDENT is in the array, NDIREC, of common block POOLTB, the data group will not be dumped by calls to DMPOOL with IFORM $>$ 0.

4. Entry point POOLPR is used whenever the data group index is not required as the data are stored.

Subroutine POOLDS (Entry point in Subroutine POOLIT)

Function: Searches for a data word stored in POOLED storage.

Common Blocks: OPTION, POOLTB

Entry Point: POOLDS

Calling Sequence: CALL POOLDS(IA,IDENT,LOCSET,LOCP)

IA	Data word to be located
IDENT	Pointer to a particular data group; set by program when PLDATA, PLPAIR, POOLDT, or POOLPR first called
LOCSET	Location returned by program (if = 0, no match has been found for the word in the specified data group; if > 0, LOCSET is the data group index of the word)
LOCP	POOLED index of the data word (set only if match is found)

Subroutine Called: ABORT

Error Messages:

Number 101	IDENT out of range in POOL search
Number 102	IDENT specified does not point to the beginning of a data group
Number 103	Improper identifier in second word of a pool record header.

Remarks:

1. If called with IDENT = 0, indicating a null data group, program returns with no action.

2. LOCP can be used with subroutines FETCH1 and STOW1 for rapid access to the POOLED data.

Subroutine POOLIT

Function: Initializes POOLED data storage parameters, purging all previously stored data.

Common Blocks: POOLTB, OPTION

Entry Point: POOLIT

Calling Sequence: CALL POOLIT

Subroutines Called: None.

Error Messages: None.

Remark:

POOLED data storage is self-initializing if it is first used by entry point PLDATA, PLPAIR, POOLDT, or POOLPR. POOLIT restores the system to this initial state.

Subroutine POOLPS (Entry point in Subroutine POOLIT)

Function: Searches for a data pair stored in POOLED storage.

Common Blocks: OPTION, POOLTB

Entry Point: POOLPS

Calling Sequence: CALL POOLPS(IA,IB,IDENT,LOCSET,LOCP)

IA,IB	Pair of data words to be located
IDENT	Pointer to a particular data group; set by program when PLDATA, PLPAIR, POOLDT, or POOLPR first called
LOCSET	Location returned by program (if = 0, no match has been found for the pair in the specified data group; if > 0, LOCSET is data group index of the first word of the pair)
LOCP	POOLED index of the first word of the pair (set only if match is found)

Subroutine Called: ABORT

Error Messages:

Number 101	IDENT out of range in POOL search
Number 102	IDENT specified does not point to the beginning of a data group
Number 103	Improper identifier in second word of a POOL record header
Number 104	Search for pair attempted on a data group with an odd number of entries

Remarks:

1. Data group must have an even number of entries to use this program.

2. If called with IDENT = 0, indicating a null data group, program returns with no action.

3. LOCP can be used with subroutines FETCH1 and STOW1 for rapid access to the POOLED data.

Subroutine PURGE (Entry point to Subroutine POOLIT)

Function: Purges a data group from POOLED storage and releases the space for future assignment.

Common Blocks: POOLTB, OPTION

Entry Point: PURGE

Calling Sequence: CALL PURGE(IDENT)

Subroutines Called: None.

Error Messages:

Number 107	The word IDENT does not point to a legal data group in POOL storage.

Subroutine STOW (Entry point to Subroutine POOLIT)

Function: Replaces data words stored in POOLED storage.

Common Blocks: POOLTB, OPTION

Entry Point: STOW

Calling Sequence: CALL STOW(IA,IDENT,LOCSET,LOCP)

IA	Data word to be stored by STOW
IDENT	Pointer to particular data group to be referenced
LOCSET	Pointer, within data group, to data word to be stored
LOCP	POOL index of the data word stored (LOCP is set by the program; if no data were previously defined for the location requested, LOCP is set to zero)

Entry Point: STOW1

Calling Sequence: CALL STOW1(IA,LOCP)

IA	Data word to be stored by STOW1
LOCP	POOL index of the data word to be stored

Subroutines Called: None.

Error Messages:

Number 106 The word IDENT does not point to a legal data group in POOL storage.

Remarks:

1. A data word must first be established using PLDATA, PLPAIR, POOLDT, or POOLPR before data can be replaced with STOW or STOW1.

2. Entry point STOW is called to store data by its sequential position in the data group (as specified by LOCSET).

3. Entry point STOW1 is called to store data using the POOL index of the data word (as specified by LOCP). The index value must have been obtained from a previous call to FETCH, STOW, PLDATA, PLPAIR, POOLDS, or POOLPS. This method of data retrieval is faster than using the STOW code and should be used when applicable.

2.2.6 File Management

FORTRAN defined files are used exclusively throughout the data generator program. For flexibility all logical unit assignments are made in the main program (with the exception of the plotting routines). As mentioned above, all generated data card images are stored on external files as generated. These card images are distributed over several files to obtain a deck in a roughly sorted order. A utility subroutine, ASSMBL, can be used to merge any number of these files onto one output file (see Section 2.6.2).

2.3 FIRST PASS AND INTERLUDE PROCESSING PROGRAMS

2.3.1 Subroutine SETUP (Initialization and Execution Control Card Processing)

Function: Permits the user to select the desired functions of the Data Generator using a freely-formatted control language. This module interprets the user's statements and sets appropriate flags in the control common block, OPTION, permitting the user to select various execution options.

Entry Point: SETUP

Calling Sequence: CALL SETUP(INPT,IPRT,IOUT)

INPT	FORTRAN logical unit number of the card reader
IPRT	FORTRAN logical unit number of the line printer
IOUT	FORTRAN logical unit number of the device on which the generated output is written

Common Blocks:

SET	Described in Section 2.6.5
OPTION	Described in Section 2.1.3

Subroutines Called: ERRMSG, PRINT, SWITCH, XRCARD

Error Messages:

Number	Level	Text
001	2	UNRECOGNIZABLE CARD

Number	Level	Text
002	2	SYNTAX ERROR ON TITLE CARD
003	2	SYNTAX ERROR ON CONTROL CARD
004	2	INVALID OPTION SPECIFIED ON CONTROL CARD
005	2	NUMBER OUT OF RANGE ON CONTROL CARD
006	3	END-OF-FILE ENCOUNTERED

Remarks:

1. Figure 2.6 shows the order of the cards processed by SETUP. All Class I cards are reproduced exactly as they appear with copies sent to both the generated output file and the printer. All Class II cards, with the exception of comment cards, are interpreted and reproduced with "$" appended at the left on the generated output file and the printer. Comment cards, which begin with a "$", are treated in the same manner as Class I cards. All cards from the input file will be listed with a running card count printed on the left.

2. Parameters not explicitly specified by Execution Control Cards are assigned the default values described in Section 2.1.3.

3. Unformatted style data cards are read with the aid of subroutine XRCARD, a NASTRAN subroutine which is described in the NASTRAN Programmer's Manual,[3] Section 3.4.19. Subroutine MASQ is library routine MASK which has been renamed to avoid name conflicts in subroutine XRCARD.

[3] "The NASTRAN Programmer's Manual," edited by Frank J. Douglas, NASA SP-223, September 1970.

First Card ⋮ "ID" card	Class I cards (not interpreted)
Control Commands, e.g., "TITLE" card ⋮	Class II cards (interpreted by data generator)
"$END" card ⋮ "BEGIN BULK" card	Class I cards (not interpreted)

Figure 2.6 - Order of Cards in Execution Control Deck

2.3.2 Subroutine INSORT (Bulk Data Card Interpreter)

Function: Controls all first pass and interlude processing of the bulk data. First pass functions performed by this subroutine include:

- Calling subroutine XINIT to initialize KEY-CHAIN storage
- Reading all bulk data cards (from card following BEGIN BULK card through ENDDATA card)
- Calling subroutine XTRACT for first pass processing of shell geometry cards
- Calling subroutine EQV to process equivalence cards
- Writing Phase I data on file NOUT2 with numeric key to data type and input card number appended (XTRACT has replaced the reference line ID's with an internal ID number on Phase I cards)
- Writing Phase II data on file NOUT3 with numeric key to data type and input card number appended (no change has been made to the reference line ID's on Phase II cards)
- Writing all bulk data cards following $END (to but not including the ENDDATA card) to file NSOUT

Interlude processing is initiated by calling subroutine XTREND.

For program development purposes the file NOUT2 is rewound and a call to subroutine RECON is made for each logical card on NOUT2. RECON reconstructs the finite element mesh density specifications from KEY-CHAIN storage reflecting assignments and changes made during the interlude. These cards are reconstructed and printed for programmer information only.

Entry Point: INSORT

Calling Sequence: CALL INSORT(INUNIT, NOUT1, NOUT2, NOUT3, NSOUT)

INUNIT	FORTRAN file number of input file containing bulk data (with right-adjusted numeric fields) - DO NOT REWIND
NOUT1	FORTRAN file number of output file on which "MAT" cards will be written
NOUT2	FORTRAN file number of output file on which processed Phase I bulk data will be written
NOUT3	FORTRAN file number of output file on which processed Phase II bulk data will be written
NSOUT	FORTRAN file number of output file for card images which are to be passed to the generated data file without processing

Common Blocks:

OPTION	Execution control parameters
RECORD	Shared storage for reading and writing bulk data
SET	Titling information for subroutine PAGE
TYPE	INDEX of card names for input and output

Subroutines Called: ABORT, EQV, ERRMSG, PRINT, RECON, SWITCH, XINIT, XTRACT, XTREND

Error Messages:

Number	Level	Text
011	2	REJECTED DATA (invalid card in BULK DATA deck)
012	3	I/O ERROR IN SUBROUTINE INSORT WHILE READING BULK DATA CARD XXX (XXX is the card number counting from the first BULK DATA card)
013	3	PROGRAM TERMINATED DUE TO ERROR COUNT IN SUBROUTINE INSORT

2.3.3 Subroutine XTRACT (Global Data Processing)

Function: Performs all first pass and interlude functions of the data generator and is called only by subroutine INSORT. KEY-CHAIN storage is initialized in the program area associated with entry point XINIT.

In the program area associated with the entry point XTRACT the first scan of a shell data card is made. Here the assignment of sequential internal identification numbers to the reference line ID's is made along with the assignment of storage space for eight boundary areas of the module being processed (four corners and four sides). A call to subroutine SPACE reserves the region of CHAIN storage required for one boundary area.

The program area associated with entry point EQV processes cards indicating the equivalence of reference line identification numbers. As these cards are encountered, the equivalence information is retained (in terms of external reference line ID's) for processing at the interlude between passes by subroutine XTREND. Processing terminates whenever a zero field is encountered or when the fifieth field has been processed.

Associated with the entry point XTREND are most of the tasks which must be completed during the interlude after the first pass through the data. This region probably contains the most tedious logic to be found in the program. The primary functions performed at this point are:

- Updating the correspondence tables between internal and external reference ID's to reflect equivalences
- Updating the KEY area to reflect equivalences
- Assigning storage space to those boundaries for which the user indicates some default value is to be used. This results in the propagation of the grid point

mesh to unspecified areas of the model.

The program area associated with entry point RECON has been added to facilitate development of the interlude processing by XTREND. After the interlude RECON operates on logical input cards, reconstructing the finite element mesh requests from KEYCHAIN storage for manual comparison with the cards as input by the user.

Common Blocks: KEYCHN, SET, OPTION, RECORD

Entry Point: EQV

Calling Sequence: CALL EQV

Entry Point: RECON

Calling Sequence: CALL RECON

Entry Point: XINIT

Calling Sequence: CALL XINIT

Entry Point: XTRACT

Calling Sequence: CALL XTRACT

Entry Point: XTREND

Calling Sequence: CALL XTREND

Subroutines Called: ERRMSG, FETCH, LOCKS, POOLPR, PRINT, PURGE, SPACE, STOW, STOW1

Error Messages:

Number	Level	Entry Point	Text
020	2	XTRACT	RL IS ZERO OR NEGATIVE OR EXCEEDS MAXIMUM NUMBER PERMITTED ON BULK DATA CARD XXX. XXX is sequence number printed with the listing of the data cards.

Error Messages (cont'd):

Number	Level	Entry Point	Text
021	2	XTREND	AN EQUIVALENCE REFERENCES THE NON-EXISTENT XXX YRL. REFERENCES IGNORED. XXX is the reference line number and Y is either Z or A indicating the orientation of the line.
022	2	XTREND	EQUIVALENCE CONFLICT AT YRL XXX AND YRL XXX. FIRST SPECIFICATION USED. XXX is a reference line number and Y is either Z or A indicating the orientation of the line.
023	2	XTREND	NON-EXISTENT RL REFERENCED IN THE XXX-TH PAIR OF INTERSECTION EQUIVALENCES. XXX is sequence number of the intersection equivalences as encountered.
024	1	XTREND	THE NUMBER OF DIVISIONS HAS NOT BEEN SPECIFIED FOR SIDE ZZZ AAA ALONG THE YRL. DEFAULT IS ONE DIVISION. ZZZ and AAA define an intersection of a ZRL and an ARL and Y is either Z or A indicating the orientation of the line.

Error Messages (cont'd):

Number	Level	Entry Point	Text
025	3	XTREND	MORE ITERATIONS ARE REQUIRED TO UPDATE CHAIN STORAGE THAN THE XXX PERMITTED. XXX is the computed theoretical maximum number of iterations which could be required to update CHAIN storage.
026	3	EQV	INTERSECTION EQUIVALENCE STORAGE EXCEEDED.

2.4 SECOND PASS PROCESSING PROGRAMS

2.4.1 Subroutine CUTUP (Process Shell Data)

Function: To generate shell data one section at a time.

Common Blocks:

KEYCHN	Storage area for module boundary data
OPTION	Contains execution control options
SET	Contains title information
REFPT	Contains the reference line numbers of the present module
TICK	Contains thicknesses and pressures on the boundary
ROD	Contains the boundary radii
PROPCE	Property identification storage for CONEND and CONENDR modules
PROPS	Property identification storage for CONE modules

Entry Point: CUTUP

Calling Sequence: CALL CUTUP

CUTUP is called only once for each set of bulk data after the data have been read in and processed.

Subroutines Called: CONE, CONEND, LOCKS, READS, REF

Method:

Subroutine READS is called to retrieve pertinent data from the input file. Subroutine REF is called nine times to transform the various combinations of the reference line numbers into usable form. Subroutine LOCKS is called four times to obtain the external reference line numbers and number of divisions for the four sides of the section and is called again four times to obtain the external reference line numbers for the corner points. The required element generation subroutine (CONE, CONEND, or CONENDR) is then called.

An appropriate CORD2C (coordinate system definition) card is generated in subroutine CUTUP along with the grid and element cards for graphic output.

Design Requirements:

Whenever new element generation modules are to be added, calls to these subroutines should be from subroutine CUTUP.

Error Messages:

Number	Level	Entry Point	Text
501	2	CUTUP	Two reference lines specified for this module have the same number---must be distinct
502	2	CUTUP	No CHAIN storage space was assigned for a module boundary
503	2	CUTUP	Illegal data (may indicate that the number of divisions along a Z-reference line exceeds the number of divisions along an A-reference line for CONEND or CONENDR modules)
504	2	CUTUP	Number of errors exceeds maximum permitted

Remark:

Any of the above error conditions will cause the generation of data for this module to be abandoned and processing to proceed to the next module.

2.4.2 Subroutine READS (Read Shell Data Card)

Function: To read in shell data from file INCARD for one section.

Common Blocks:

OPTION	Contains execution control options
ROD	Contains the boundary radii
PHIA	Contains input data
REFPT	Contains the reference line numbers of the present module
TICK	Contains thicknesses and pressures on the boundary
PROPS	Property identification storage for CONE modules
PROPCE	Property identification storage for CONEND and CONENDR modules

Entry Point: READS

Calling Sequence: CALL READS(ISHELL, ZL1, ISTART, ILAST, INCARD)

ISHELL	Indicates the module type If 1, CONE section If 3, CONEND section If 5, CONENDR section
ZL1	Length of the section
ISTART	Sequence number of the current section
ILAST	Read indicator. 0 indicates continue reading records from file INCARD; 1 indicates last record to be read from file INCARD

Subroutines Called: PRINT

Method:

Read shell data from file INCARD including module type, reference line numbers, slopes of the azimuthal lines, length, angular

width of section, and radius and thickness at each corner. Slopes of the azimuthal lines are converted to radians. If the last record is being read, then ILAST is set to 1.

Error Messages: None.

2.4.3 Subroutine TERP (Linear Interpolation Routine)

Function: To perform an averaged two-dimensional linear interpolation within a structural module.

Common Blocks: None.

Entry Point: TERP

Calling Sequence: X = TERP(A, B, C, D, T1, T, Z1, Z)

X	Interpolated result
A	Property at point A
B	Property at point B
C	Property at point C
D	Property at point D
T1	Angular spacing or the difference in the azimuthal coordinates of two adjacent grid points
T	Azimuthal coordinate of the interpolated point
Z1	Axial spacing or the difference of the axial coordinates of two adjacent grid points
Z	Axial coordinate of the interpolated point

Method:

$$TERP = .5\left(P_1 + (P_2 - P_1)\frac{Z1}{Z} + P_3 + (P_4 - P_3)\frac{T1}{T}\right)$$

where
$$P_1 = A + (D - A)\frac{T1}{T}$$

$$P_2 = B + (C - B)\frac{T1}{T}$$

$$P_3 = A + (B - A)\frac{Z1}{Z}$$

$$P_4 = D + (C - D)\frac{Z1}{Z}$$

Error Messages: None.

Remark: TERP is used to find the radius at an internal grid point, i.e., a grid point not on the boundary, and is also used to find the thickness and the pressure loading of individual elements of a module.

2.4.4 Subroutine CONE (CONE Module Processing)

Function: To set up the physical dimensions of the CONE module.

Common Blocks:

KEYCHN	Storage area for module boundary data
OPTION	Contains execution control options
PHIA	Contains input parameters
REFPT	Contains the reference line numbers of the present module
ROD	Contains the boundary radii

Entry Point: CONE

Calling Sequence: CALL CONE(ZL1, DIVZ, DIVA, DIVZ2, DIVA2)

ZL1	Length of the projection of the module on the Z-axis
DIVZ	Number of divisons along side 1
DIVA	Number of divisions along side 2
DIVZ2	Number of divisions along side 3
DIVA2	Number of divisions along side 4

Subroutines Called: LOCKS, QUADS, TRI

Method:

Subroutine LOCKS is called to check whether the coordinates of the corner points have been calculated. Coordinates already calculated will override the input specifications.

The radius at B, r_{bi}, will be calculated using either Equation (1) or Equation (2).

$$r_{bi} = r_{ai} + ZL1 \tan(\phi_i - \frac{\pi}{2}) \text{ for } \phi_i > \frac{\pi}{2}\text{ , } i = 2 \text{ or } 4 \qquad (1)$$

$$r_{bi} = r_{ai} - ZL1 \tan(\frac{\pi}{2} - \phi_i) \text{ for } \phi_i < \frac{\pi}{2}\text{ , } i = 2 \text{ or } 4 \qquad (2)$$

where r_{ai}, length ZL1, and ϕ_i are specified as input. By rewriting Equations (1) or (2), the length ZL1 of the section may be calculated

only if ϕ_i, r_{ai} are given as input or if r_{ai} and r_{bi} are known from previous calculations. When $\phi_i = 0^o$ or $\phi_i = 180^o$, the user must specify the radii, r_{ai} and r_{bi}, and then the length ZL1 will be set to zero.

Error Messages:

Number	Level	Entry Point	Text
520	2	CONE	Calculated length ZL1 and ZL2 do not agree; value of ZL1 will be used
521	2	CONE	$\lvert DIVZ-DIVZ2 \rvert \not< \min(DIVA, DIVA2)$; will continue to next module
522	2	CONE	$\lvert DIVZ-DIVA2 \rvert \not< \min(DIVZ, DIVZ2)$; will continue to next module
523	2	CONE	Another radius must be given as input, R_{ai} or R_{bi}, because $\phi_i = 0^o$ or $\phi_i = 180^o$; will continue to next module

Remarks:

1. If DIVZ $\neq$ DIVZ2 and DIVA $\neq$ DIVA2, then the relations $\lvert DIVZ-DIVZ2 \rvert < \min(DIVA, DIVA2)$ and $\lvert DIVA-DIVA2 \rvert < \min(DIVZ, DIVZ2)$ must be satisfied.

2. Geometry of the CONE module is shown in Figure 1.11.

2.4.5 Subroutine QUADS (Frustum of Cone Generation - Varying Mesh)

Function: To generate a finite element mesh for CONE modules bounded by four reference lines.

Common Blocks:

GRIDPT	Grid point numbering storage for present module
KEYCHN	Storage area for module boundary data
OPTION	Contains execution control options
REFID	Contains internal reference line number information
REFPT	Contains the reference line numbers of the present module
BREFID	Contains external and corner line number information
TICK	Contains thicknesses and pressures on the boundary
PROPS	Property identification storage for CONE modules
TYPE	Index of card names for input and output
PROP	Constant data for GRID and connection cards
ROD	Constant boundary radii
SET	Contains title information
PROPCE	Property identification storage for CONEND and CONENDR modules
PHIA	Contains input parameters

Entry Point: QUADS

Calling Sequence: CALL QUADS(R1,T1,Z1,THETA1,ZL,IDIVA, IDIVZ,TTHETA,ZL1,IDIVA2,IDIVZ2,IQ, TT,I8,I12,I14)

R1,T1,Z1	Coordinates, in a cylindrical coordinate system, locating the grid point at A of this cone module
THETA1	$\Delta\theta$, angular increment along the longitudinal reference lines

ZL	ΔZ, increment along the azimuthal reference lines
IDIVA	Number of divisions along side 2
IDIVZ	Number of divisions along side 1
TTHETA	Azimuthal width
ZL1	Length of the cone module projected on the Z-axis
IDIVA2	Number of divisions along side 4
IDIVZ2	Number of divisions along side 3

(Figure 1.11 shows the relationships of the sides of the cone module.)

IQ   Option flag;

If 0, cone module is composed of all quadrilateral elements

If 1, cone module is composed of triangular elements on the upper part and quadrilateral elements on the lower part (Figure 2.7)

If 2, cone module is composed of quadrilateral elements on the upper part and triangular elements on the lower part (Figure 2.8)

If 3, cone module is composed of only triangular elements on the upper part and both quadrilateral and triangular elements on the lower part (Figure 2.9 a & b)

If 4, cone module is composed of both triangular and quadrilateral elements on the upper part (generated by subroutine TRI) and only triangular elements on the lower part (QUADS will generate these elements) (Figure 2.9 c & d)

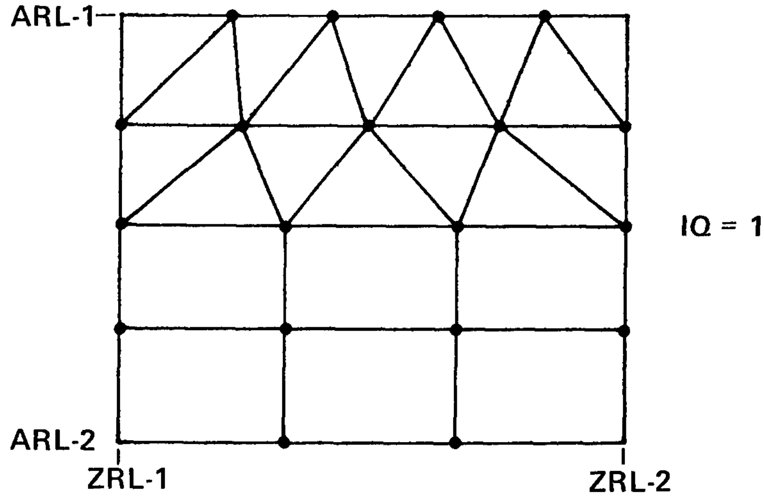


Figure 2.7 – CONE Module with Varying Mesh along the First Azimuthal Reference Line (ARL 1)

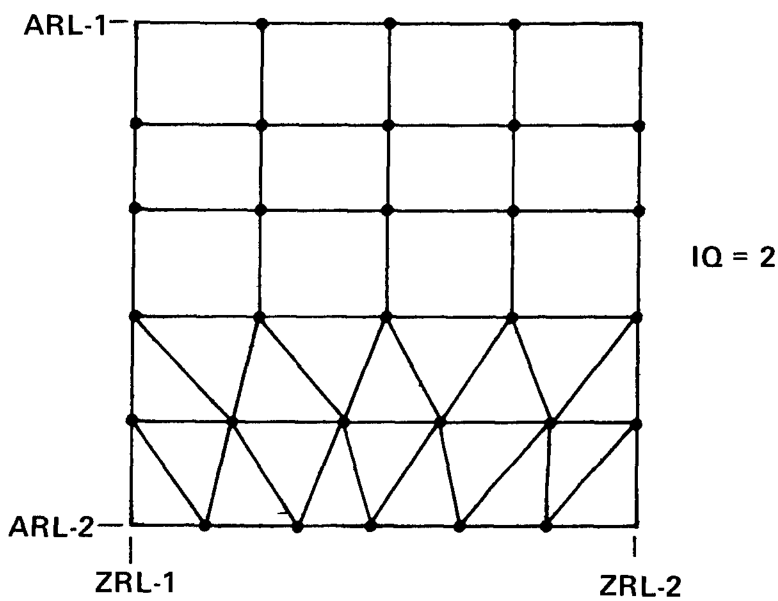


Figure 2.8 – CONE Module with Varying Mesh along the Second Azimuthal Reference Line (ARL 2)

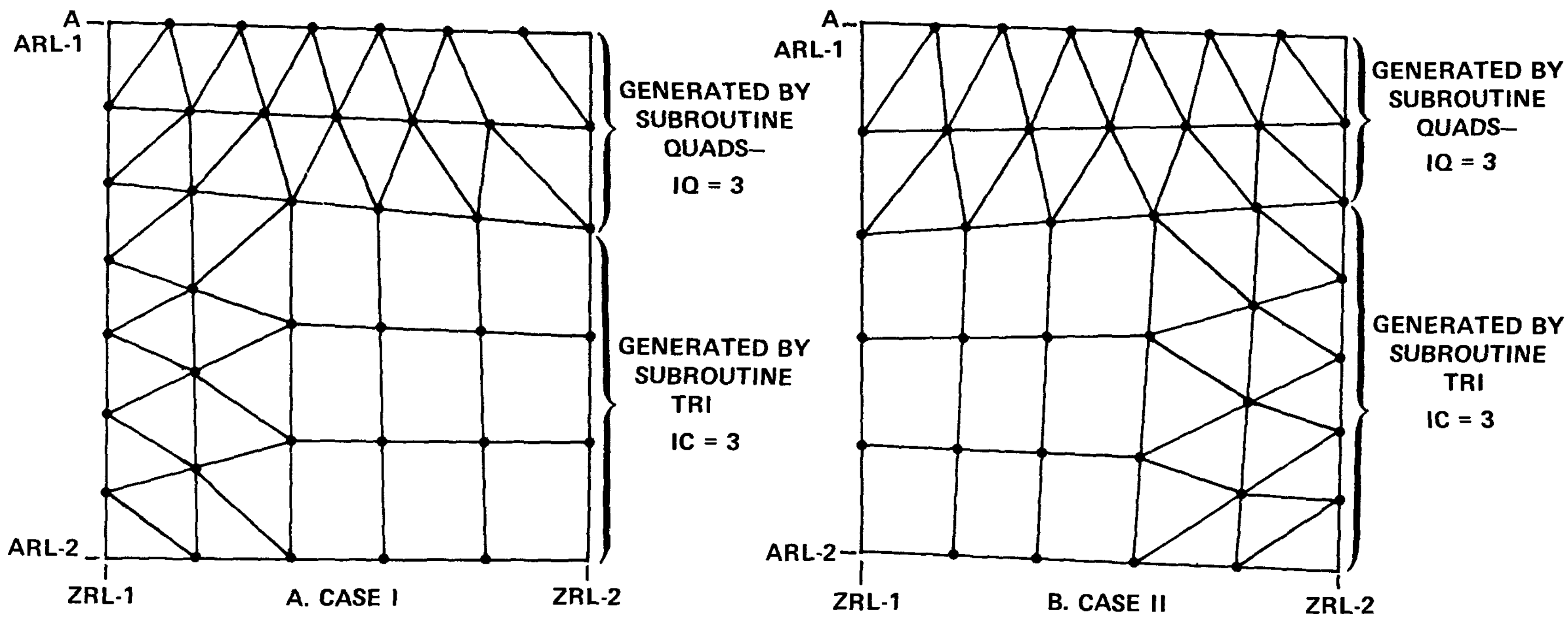


Figure 2.9 - CONE Module with Varying Mesh along All Four Boundaries

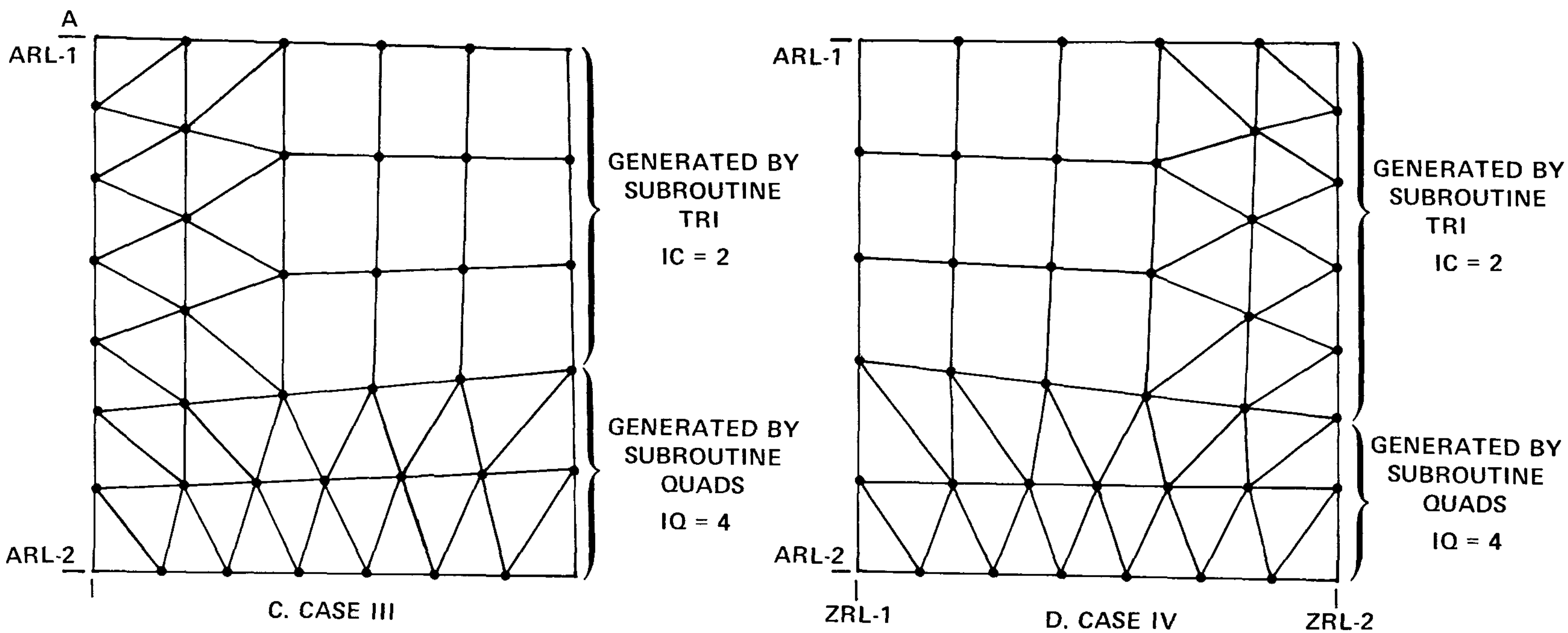


Figure 2.9 - CONE Module with Varying Mesh along All Four Boundaries

TT	T1 value of the coordinates (R1, T1, Z1) of the gridpoint at A of the cone module. This variable is used only if IQ=4 and the subroutine is called from subroutine TRI or if IQ=3; otherwise it is set to zero (0)
I8	File containing grid cards
I12	File containing connection cards
I14	File containing pressure loading cards and property identification cards

Subroutines Called: CEPROP, LOCKS, PRINT, PROPER, TERP, TRI

Method:

This subroutine divides a CONE module bounded by four reference lines into quadrilateral elements when a uniform mesh is required or into a combination of triangular and quadrilateral elements when a change in the mesh density is needed. The value of the variable IQ indicates which type of mesh will be generated.

Each grid point is numbered starting at the top of the CONE module, going from left to right, then from top to bottom according to the 7-digit numbering convention described in Section 1.3.1.

The coordinates of each grid point are calculated as the grid point is numbered. Coordinates of grid points on the boundary of the CONE module are stored for future reference by calling subroutine LOCKS. Then GRID cards are printed and stored on file I8. After all the grid points are numbered for the section, the elements are defined by calling subroutine PROPER or CEPROP to obtain property ID's and thicknesses and CQUARD2 or CTRIA2 cards are printed and stored on file I12. Pressure loading cards (PLOAD) for each element are printed and stored on file I14.

The ID of the grid point at the top left of the element is used as the element identification of quadrilaterals. The identification of

triangular elements is obtained by starting with the node number of the top left grid point and numbering the triangles in the Z-direction, incrementing each (fourth digit) by one. An example of the element and grid point numbering generated by QUADS is shown in Figure 1.6.

Error Messages: None.

2.4.6 Subroutine <u>PROPER</u> (Element Property Generation - Quadrilateral Elements)

Function: To set up property identification cards for homogeneous quadrilateral elements.

Common Blocks:

TYPE	Index of card names for input and output
PROPS	Property identification storage for CONE modules
SET	Contains title information

Entry Point: <u>PROPER</u>

Calling Sequence: CALL PROPER(LELE, THKM, I14)

LELE	Number of quadrilateral elements in the section
THKM	Array containing the thickness of each quadrilateral
I14	File containing property identification cards

Subroutine Called: PRINT

Method:

A sequential property identification number (PID) is assigned to each unique thickness associated with quadrilateral elements. Thicknesses for elements in this section are stored in column 2 of the array PIDT and the corresponding PID's are stored in column 1. PQUAD2 cards are printed and stored on file I14 whenever a new PID is encountered. The array IPID, in common block PROPS, will contain the PID of each element in the order in which the thicknesses are given in array THKM.

Error Messages: None.

2.4.7 Subroutine CONEND (CONEND and CONENDR Modules - Geometry Processing)

Function: To set up the physical dimensions of the element types, CONEND and CONENDR.

Common Blocks:

KEYCHN	Storage area for module boundary data
PHIA	Contains input parameters
ROD	Contains boundary radii
REFPT	Contains the reference line numbers of the present module

Entry Point: CONEND

Calling Sequence: CALL CONEND(ZL1, DIVZ, DIVA, DIVZ2, DIVA2, ICONED)

ZL1	Length of the element type
DIVZ	Number of divisions along side 1
DIVA	Number of divisions along side 2
DIVZ2	Number of divisions along side 3
DIVA2	Number of divisions along side 4
ICONED	Option flag; 0 indicates a CONEND section 1 indicates a CONENDR section

Subroutines Called: LOCKS, TRI

Method:

The cone-shaped end module is bounded by four reference lines, one side being a null side. The geometrys of the CONEND and CONENDR modules are shown in Figures 1.12 and 1.13. Notice that the apex is associated with side 1 or side 3 depending on the selection of CONEND or CONENDR.

Subroutine LOCKS is called to determine whether the coordinates of the corner points, A, B, and D have been calculated. Those coordinates that have been calculated will override the input specifications.

The length L will be calculated using Equation (3)

$$L = R \tan \phi \qquad (3)$$

where R and ϕ are specified input. Note that $R = R_a$ for CONENDR modules and $R = R_b$ for CONEND modules. The radius R will be calculated from Equation (3) if L and ϕ are specified and $\phi \neq 0^{\circ}$.

Error Messages:

Number	Level	Entry Point	Text
530	2	CONEND	Calculated lengths for CONEND do not agree; value of ZL1 is used.
531	2	CONEND	Calculated radius does not equal radius already punched; value of punched radius is used.

2.4.8 Subroutine TRI (CONEND and CONENDR Modules - Mesh Generation)

Function: To generate a finite element mesh for CONEND or CONENDR modules and for CONE modules with varying mesh density.

Common Blocks:

GRIDPT	Gridpoint numbering storage for present module
KEYCHN	Storage area for module boundary data
OPTION	Contains execution control options
REFID	Contains internal reference line number information
REFPT	Contains the reference line numbers of the present module
BREFID	Contains external and corner reference line number information
TICK	Contains thicknesses and pressures on the boundary
TYPE	Index of card names for input and output
PROPCE	Property identification storage for CONEND and CONENDR modules
PROPS	Property identification storage for CONE modules
PROP	Constant data for GRID and connection cards
PHIA	Contains input parameters
ROD	Contains boundary radii
SET	Contains title information

Entry Point: TRI

Calling Sequence: CALL TRI(R1,T1,IDIVA,IDIVZ,DTHETA,ZL1, IDIVZ2,IDIVA2,IC,TT,I8,I12,I14)

R1,T1,Z1	Coordinates, in a cylindrical coordinate system, of the grid point at A of the cone end module that is to be subdivided into regions of quadrilateral and triangular elements
IDIVA	Number of divisions along size 2
IDIVZ	Number of divisions along side 1

DTHETA	Azimuthal width
ZL1	Length of cone end module projected on the Z-axis
IDIVZ2	Number of divisions along side 4
IDIVA2	Number of divisions along side 3
IC	Option flag; If 0, indicates a CONEND module If 1, indicates a CONENDR module If 2, a cone module is composed of both triangular and quadrilateral elements on the upper part and only triangular elements on the lower part; Figure 2.9 c & d If 3, cone end type subdivision of a cone module composed of only triangular elements on the upper part (subroutine QUADS will be used to generate these elements) and both triangular and quadrilateral elements on the lower part (subroutine TRI will be used to generate these elements); Figure 2.9 a & b
TT	T1 value of the coordinate (R1, T1, Z1) of the grid point at A of the cone module; this variable is used only if IC=3 and this subroutine is called from subroutine QUADS; otherwise it is set to zero
I8	File containing grid cards
I12	File containing connection cards
I14	File containing pressure loading and property ID cards

Subroutines Called: CEPROP, LOCKS, PRINT, PROPER, QUADS, TERP

Method:

This subroutine is used to generate an idealization for CONEND, CONENDR, and CONE modules. For CONEND and CONENDR modules which are bounded by three reference lines, the idealization may consist of either all triangular elements or a combination of triangular and quadrilateral elements, depending on the grid point density along the reference lines. This subroutine may also be used to generate idealizations for CONE modules which are bounded by four reference lines, requiring both triangular and quadrilateral elements to achieve the correct mesh density. The value of the variable IC indicates which type of mesh configuration is required.

The grid point and element identification numbers follow the 7-digit convention described in Section 1.3.1.

Error Messages: None.

2.4.9 Subroutine CEPROP (Element Property Generation - Triangular Elements)

Function: To set up property identification for homogeneous triangular elements.

Common Blocks:

TYPE	Index of card names for input and output
PROPCE	Property identification storage for CONEND and CONENDR modules
SET	Contains title information

Entry Point: CEPROP

Calling Sequence: CALL CEPROP(LK, THKM, I14)

LK	Number of triangular elements in the section
THKM	Array containing the thickness of each triangular element in the section
I14	File containing pressure loading cards and property identification cards

Subroutine Called: PRINT

Method:

A sequential property identification number (PID) is assigned to each unique thickness associated with triangular elements. These thicknesses are stored in column 2 of the array PID and the corresponding PID's are stored in column 1. PTRIA2 cards are printed and stored on file I14 whenever a new PID is encountered. The array, JPID in common block PROPCE, will contain the PID of each element in the order in which the thicknesses appear in the array THKM.

Error Messages: None.

2.4.10 Subroutine REF (Identification Number Generation)

Function: To build a unique 7-digit identification number from two or four reference line numbers.

Common Blocks:

REFID	Contains internal reference line number information
BREFID	Contains external and corner line number information
KEYCHN	Storage area for module boundary data

Entry Point: REF

Calling Sequence: CALL REF(IRID,IA1,IZ1,IA2,IZ2)

IRID n^{th} call to REF (where n = 1 to 9)

- If 1, indicates internal numbering
- If 2, indicates boundary numbering for side 1 (see Figure 1.11 for relationship of sides)
- If 3, indicates boundary numbering for side 2
- If 4, indicates boundary numbering for side 3
- If 5, indicates boundary numbering for side 4
- If 6, indicates numbering for corner A (see Figure 1.11)
- If 7, indicates numbering for corner B
- If 8, indicates numbering for corner C
- If 9, indicates numbering for corner D

IA1	First A-reference line number
IZ1	First Z-reference line number
IA2	Second A-reference line number
IZ2	Second Z-reference line number

Subroutine Called: PRINT

Method:

Subroutine REF prepares the reference line numbers for the 7-digit numbering scheme for the corner grid points and for the boundaries as described in Section 1.3.1. The seventh digit (leftmost) is found for each corner point and for the boundaries, and the values are stored in

the COMMON BLOCK BREFID. The reference line numbers and the corresponding seventh digit for internal numbering scheme are stored in the COMMON BLOCK REFID.

Error Messages: None.

Remark:

Parameters IA2 and IZ2 are used only when IRID=1. For other calls they must be dummy variables or constants.

2.5 NASPL - GRAPHICAL OUTPUT PROCESSING

The program NASPL* was inserted as part of the data generator to provide graphics output. The main program of NASPL became the main program of the second primary level in the overlay structure. The subroutine GOOGAN was omitted since the generated data are already right adjusted.

Options for NASPL are selected by setting the parameters in the common block PLOT1. The following values are currently specified for plots of generated data:

ITYPE = 1	Generates both perspective and orthogonal plots
I2D = 2	Plots two-dimensional elements only for orthogonal plots
IXYZ = 4	Generates orthogonal plots in all three viewing planes, xy, yz, and xz
I3D = 2	Plots two-dimensional elements only for perspective plots
IPRINT = 0	No grid points or coordinate information to be printed
JPRINT = 0	No element information to be printed
TH1 = 0.0	First angle of rotation for calculation of point of view for perspective plotting
TH2 = 0.0	Second angle of rotation for calculation of point of view for perspective plotting

With the current implementation only the generated data will be plotted and any manually prepared data will be ignored by NASPL.

---

* NASPL is an in-house plotting program used for NASTRAN data checking at the Naval Ship Research and Development Center.

Subroutine PLASSM is used to merge the data onto the plot data file and to delete any comment cards inserted by the generator. The exclusion of manually prepared data and comment cards is necessary for correct operation of this abbreviated version of NASPL.

2.6 UTILITY PROGRAMS

The following collection of subroutines is available to the programmer for general housekeeping tasks and may be used as required throughout the program.

2.6.1 Subroutine ABORT (Data Storage Dump Routine)

Function: Dumps selected common storage areas for debugging purposes.

Common Blocks: OPTION: See Section 2.1.3

Entry Point: ABORT

Calling Sequence: CALL ABORT

Subroutines Called: SECOND, PRINT, DMKYCH, DMPOOL

Error Messages: None.

Remarks:

1. Programmers may add dumps of regions of interest either directly or through subroutine calls.

2. The current processing step in the program is determined from the seventh word of the OPTION common block and only those areas which are meaningful at that stage will be dumped.

3. Current CPU time for the job and the CPU time since the last call to ABORT will be printed with each call.

2.6.2 Subroutine <u>ASSMBL</u> (File Merging Routine)

Function: Merges several files containing card images onto one file and writes an "ENDDATA" card at the end of the merged file.

Entry Point: <u>ASSMBL</u>

Calling Sequence: CALL ASSMBL(NSOUT,NFILES,MFILES)

NOUT	FORTRAN logical file number for the merged file
NFILES	Number of files to be merged
MFILES	Array containing the FORTRAN logical file numbers of the files to be merged, in the order that they are to appear on the merged file

Entry Point: <u>PLASSM</u>

Calling Sequence: CALL PLASSM(NSOUT,NFILES,MFILES)

Subroutines Called: None.

Error Messages: None.

Remarks:

1. All files except the merged file (NSOUT) are rewound both before and after ASSMBL processing.

2. PLASSM deletes all cards beginning with a dollar sign (NASTRAN COMMENT cards) as the file is assembled.

2.6.3 Subroutine ERRMSG (Error Message Printer)

Function: Writes a message header on the printed output file.

Common Blocks: OPTION

Entry Point: ERRMSG

Calling Sequence: CALL ERRMSG(IFATAL, NUM)

IFATAL Level of severity of error

If 1, non-fatal

If 2, fatal; will not continue into the analysis phase after generation

If 3, fatal; will stop at once

NUM Message number

Subroutine Called: PRINT

Error Messages: None.

Remarks:

1. If error level is 1, the following message will be printed after one line is skipped:

*******bbXXXYZZZb ,

where XXX is the current step number as indicated by the seventh word of the OPTION common block, Y is the value of IFATAL, and ZZZ is the error number. For error levels of 2 or 3, the following message will be printed after one line is skipped:

*FATAL*bbXXXYZZZb ,

where XXX, Y, and ZZZ are defined as above.

2. ERRMSG will set the NSR word of the OPTION common block to the value of IFATAL if this value is greater than the current NSR value.

3. ERRMSG calls subroutine PRINT to properly update page and line count information. For messages which are longer than one line, subroutine PRINT should be called to reflect additional lines printed.

4. To issue an error message, call ERRMSG and then write a one-line message, preceding the message text with a plus sign and

seventeen blank spaces, e.g.,

```
      CALL ERRMSG(1,99)
      WRITE(6,990) KOUNT
990   FORMAT(1H+,17X,*message text*,I5).
```

2.6.4 Subroutine GOOGAN (NASTRAN Format Translator)

Function: To convert data card images from NASTRAN bulk data format to FORTRAN acceptable format.

Common Blocks: SET: See Section 2.6.5

Entry Point: GOOGAN

Calling Sequence: CALL GOOGAN(KQ, KB, NIN, NOUT)

KQ = -1	For data generator applications (processes bulk data cards only)
KB = 2	Data field lengths to be left unchanged after conversion
NIN	FORTRAN logical file number for the file from which the original deck is read
NOUT	FORTRAN logical file number for the file on which the converted deck is written

Subroutines Called: PRINT

Error Messages: None.

Remarks:

1. NASTRAN BULK DATA cards are subdivided into ten 8-column fields. On each card fields 2 through 9 may contain numeric information placed anywhere within the field as long as there are no imbedded blanks (except before exponents) and no decimal points included in integer numbers, and as long as decimal points are included with real numbers. This program right adjusts the number within a field so that it may be read as a real number using E8.0 or F8.0 FORTRAN format specifications.

2. This program lists each card after processing along with a sequence number corresponding to its position in the deck.

2.6.5 Subroutine PRINT (Heading and Page Control Routine)

Function: To control the pagination of printed output including the printing of a banner page; the printing of page headings with the problem title, date, and page number; and the printing of titles for output quantities.

Common Blocks:

OPTION: See Section 2.1.3

SET: Length 54 words

Words	Description
1-8	Problem title, set by SETUP, 10 characters per word
9-20	Unused on CDC 6700
21	Current date
22-23	Unused on CDC 6700
24	Page count
25-54	Data title, set by calling program - 4 characters left justified in each word

Calling Sequence: CALL PRINT(LINES)

If LINES = 0	Prints banner page and resets page count to zero
> 0	If LINES printed lines will fit on the current page, then the line count is incremented by LINES and control is returned to the calling program. If LINES printed lines will not fit on the current page, then the program skips to a new page, increments the page count by one, prints the heading and data title, and sets the line count to LINES+5.
< 0	The program skips to a new page, increments the page count by one, prints the heading and data title, and sets the line count to 4-LINES.

Error Messages: None.

Remarks:

1. LINES is the number of lines of output which will be printed following this call, if LINES is positive. When LINES is negative, -(LINES-1) lines will be printed.

2. The use of a negative value for LINES forces the beginning of a new page. If the first character of the data title is non-blank, that character is changed to a blank and that title will be used on all pages until PAGE is again called with a negative argument. If the first character is blank, the complete data title is changed to blanks.

3. The number of lines to be printed on each page is stored in the fifteenth word in common block OPTION.

2.6.6 Function Subprogram SWITCH (Character Manipulation Routine)

Function: To construct a word by substituting the first (leftmost) character of one word into a specified character position of another word.

Entry Point: SWITCH

Calling Sequence: D = SWITCH(A, I, B)

D	Constructed word
A	Pattern word
I	Position of the character to be changed in pattern word (integer, $1 \le I \le 10$)
B	Replacement character (only leftmost six bits used)

Subroutines Called: ABORT, ERRMSG

Error Messages:

Number	Level	Entry Point	Text
30	2	SWITCH	SUBSTITUTION CHARACTER OUT OF RANGE IN "SWITCH" FUNCTION

Remarks:

1. This routine is for CDC 6000 series computers only.
2. Characters are six bits long. The first character is defined to be the leftmost six bits of a word.
3. A, B, and D may all refer to the same word in memory.

3. PROGRAM MESSAGES

Program messages will be preceded by seven asterisks for non-fatal errors, or by "*FATAL*" for fatal errors, and a message code. The message code has the form XXXYZZZ, where XXX is the current processing step, Y is the severity level of the highest severity encountered thus far, and ZZZ is the message number (leading zeros will not be printed with this number).

Number	Level	Subroutine (Entry Point)	Remarks
001	2	SETUP	Unrecognizable card
002	2	SETUP	Syntax error on title card
003	2	SETUP	Syntax error on control card
004	2	SETUP	Invalid option specified on a control card
005	2	SETUP	Number out of range on a control card
006	3	SETUP	End-of-file encountered while reading card input
011	2	INSORT	Invalid card in BULK DATA deck
012	3	INSORT	I/O error occurred while reading BULK DATA card
013	3	INSORT	Program terminated due to error count
020	2	XTRACT	Zero or negative reference line number specified or maximum number of reference lines has been exceeded
021	2	XTRACT (XTREND)	Equivalence specifies a non-existent reference line --- equivalence ignored
022	2	XTRACT (XTREND)	Listed equivalence conflicts with earlier specification --- equivalence ignored

Number	Level	Subroutine (Entry Point)	Remarks
023	2	XTRACT (XTREND)	Intersection equivalence specifies a non-existent reference line --- equivalence ignored
024	1	XTRACT (XTREND)	The number of divisions has not been specified along the edge of a module --- default assignment will be used
025	3	XTRACT (XTREND)	Interlude processing requires more iterations to propagate mesh specifications than the theoretical maximum --- program error
026	3	XTRACT (EQV)	Intersection equivalence storage exceeded. Reduce the number intersection equivalence or increase POOL storage
030	2	SWITCH	A request has been made to substitute a character which cannot be processed. A maximum of ten characters will be considered as word for SWITCH processing.
040	1	DATGEN	Generation and plotting complete for one data case
041	3	DATGEN	This job has been terminated due to the occurrence of a Level 3 error (or a Level 2 error when "NASTRAN=YES" has been specified)
050	2	LOCKIT (LOCKIT) (LOCKS)	CHAIN storage has been exceeded. This usually results from attempting to generate models with too many gridpoints on module boundaries. It may also result from use of too many modules which employ higher level (levels greater than 1) storage for parameter communication. In the first case the problem size probably should be

Number	Level	Subroutine (Entry Point)	Remarks
			reduced. In the second case the problem could be split into two or more segments.
051	2	SPACE (SPACE)	Two adjacent modules define a different number of gridpoints for a common boundary. The first specification will be used.
100	2	POOLIT (POOLPR) (POOLDT)	POOL storage exceeded during data insertion operations
101	2	POOLIT (POOLPS) (POOLDS)	Illegal identification in POOL search
102	2	POOLIT (POOLPS) (POOLDS)	IDENT specified does not point to a set
103	2	POOLIT (POOLPS) (POOLDS)	Illegal second identifier in POOL record
104	2	POOLIT (POOLPS)	A set of pairs has an odd number of entries
105	2	POOLIT (FETCH) (FETCH1)	Same as 101 or 102
106	2	POOLIT (STOW) (STOW1)	Same as 101 or 102
107	2	POOLIT (PURGE)	Same as 101 or 102
108	2	POOLIT (POOLPR) (POOLDT)	Illegal IDENT in POOL storage operation
501	2	CUTUP	Two parallel reference lines have the same ID in the specifications for this module. Program continues to process next section.

Number	Level	Subroutine (Entry Point)	Remarks
502	2	CUTUP QUADS	No storage space has been assigned for an edge of this module. Program continues to process the next section.
503	2	CUTUP	Number of divisions along Z-reference line is greater than the number of divisions along the A-reference line. Error in input data for CONEND or CONENDR module. Program continues to process the next section.
504	3	CUTUP	The number of errors encountered during one processing step has exceeded the maximum permitted. (This limit is specified in the 14th word of the OPTION COMMON block.)
520	2	CONE	Calculated lengths ZL1 and ZL2 do not agree. Value of ZL1 will be used.
521	2	CONE	$\|DIVZ-DIVZ2\|$ is not less than min(DIVA, DIVA2). Program continues to the next section.
522	2	CONE	$\|DIVA-DIVA2\|$ is not less than min(DIVZ, DIVZ2). Program continues to the next section.
523	2	CONE	Another radius must be specified for this module, either R_{ai} or R_{bi}, since $\phi_i = 0^o$ or $\phi_i = 180^o$. Program continues to the next section.
530	2	CONEND	Calculated lengths for CONEND or CONENDR do not agree. The value of ZL1 is used. Program continues to generate data.
531	2	CONEND	Calculated radius does not agree with radius already punched. Value of radius punched is used.

4. SAMPLE PROBLEMS

4.1 DEMONSTRATION CONE

The structure shown in Figure 4.1 was idealized as 15 data generator modules. The three modules at the closed end of the structure (1, 2, and 3 in Figure 4.1) are the CONEND type; all others are CONE type modules. For reference, a listing of the user supplied data, selected pages of program messages and generated data, and three structural plots from this run have been included. The following remarks apply to the circled numbers throughout the sample pages.

Remarks:

1. Default specifications have been chosen for all options except page titling and structure plotting. Note that the dollar signs preceding the TITLE and PLOTID cards have been added by the program.

2. Execution control cards have been included which are to be passed directly to NASTRAN.

3. Note Z-equivalence specification for the intersection of cylinder and cone portions of the structure.

4. All input quantities have been right-justified within each field.

5. Examples of geometric specifications which will be propagated among the modules as the idealization is generated.

6. Example of data generated by a CONEND module.

7. Only quantities appearing explicitly on data cards will be listed in this section.

8. The relationship between the user's reference line numbers and generated data is indicated in this section.

9. The radii used by the module are listed, whether they appear explicitly on the data card or not.

10. Generated data cards are listed as produced. Because some modules are internally divided into subregions (as this one is), headings

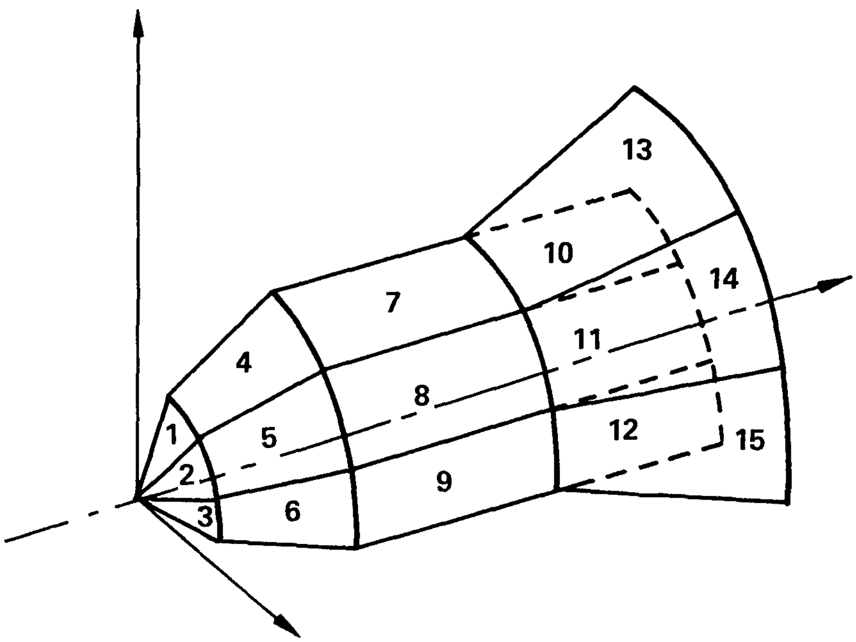


Figure 4.1 – Demonstration Cone

indicating type of card may not always be appropriately placed.

11. Example of data generated by the CONE module.

12. Only one property card is generated for each combination of thickness, material identification number, and element type encountered during the run.

13. The structure plotter plots (and thus counts) only generated elements and grid points.

14. Axes appearing in the plots are in NASTRAN's basic rectangular coordinate system and not in the generated coordinate system.

15. Grid point numbers are printed on the structural plots if there is sufficient clear area for the numbers to be legible among the element plots.

16. The scale is calculated for each view so that the structure will always fill the plotting frame.

```
CARD COUNT              E X E C U T I O N   C O N T R O L   D E C K

   1         ID  MCKEE  DEMO  CONE
   2     1  $TITLE =     D E M O N S T R A T I O N   C O N E . . .
   3        $PLOTID = MCKEE, CODE1844, EXT71493
   4         $END
   5     2  CEND
   6         TITLE=NASTRAN TITLE
   7         DISP(PUNCH)=10
   8         BEGIN BULK
```

CARD COUNT   I N P U T   D A T A   D E C K

1	ZEQU	8	7					
2	CONEND	1	1	3	2	35.		30.+001
3	+001	3.	.1	1.	2	1		+002
4	+002				4			
5	CONEND	1	2	3	3			30.+003
6	+003		.1	1.	2	1		
7	CONEND	1	3	3	4			30.+005
8	+005		.1	1.	2	1		
9	CONE	3	1	5	2	125.		+007
10	+007		.1	1.		1		+008
11	+008	5.			3			
12	CONE	3	2	5	3			+009
13	+009		.1	1.		1		
14	CONE	3	3	5	4			+010
15	+010		.1	1.		1		
16	CONE	5	1	7	2	90.	6.	+011
17	+011		.1	1.		1		+012
18	+012				5			
19	CONE	5	2	7	3			+013
20	+013		.1	1.		1		
21	CONE	5	3	7	4			+014
22	+014		.1	1.		1		
23	CONE	7	1	9	2	90.	7.5	+015
24	+015		.1	1.	4	1		+016
25	+016				7			
26	CONE	7	2	9	3			+017
27	+017		.1	1.	4	1		
28	CONE	7	3	9	4			+018
29	+018		.1	1.	4	1		
30	CONE	8	1	11	2		7.5	+019
31	+019		.1	1.		1		+020
32	+020	8.			4			+021
33	+021				2			
34	CONE	8	2	11	3			+022
35	+022		.1	1.		1		+023
36	+023							+024
37	+024				2			
38	CONE	8	3	11	4			+025
39	+025		.1	1.		1		+026
40	+026							+027
41	+027				3			
42	ENDDATA							

```
D A T A   G E N E R A T O R   O U T P U T   D E C K
(6)——SECTION   1

         DEFINED BY USER
       LENGTH OF SECTION (L)=  -0.0000

(7)——  AZIMUTHAL ANGLE (THETA)=  30.0000

       SLOPE OF ARL1 (PHI1)=   35.000

       SLOPE OF ARL2 (PHI2)=   -0.000

             THE GRID POINT NUMBER AT THE INTERSECTION OF Z1,A1( 1, 1) IS     100100

(8)——        THE GRID POINT NUMBER AT THE INTERSECTION OF Z2,A1( 3, 1) IS     200100

             THE GRID POINT NUMBER AT THE INTERSECTION OF Z1,A2( 1, 2) IS     100100

             THE GRID POINT NUMBER AT THE INTERSECTION OF Z2,A2( 3, 2) IS     200200

(9)——THE RADIUS AT EACH CORNER... RA=  0.00 RB=   3.00 RC=   3.00 RD=   0.000
```

10 — D A T A   G E N E R A T O R   O U T P U T   D E C K

SECTION BOUNDED BY ZRPS=   1   3 AND ARPS   1   2

*****GRID CARDS

GRID	ID	CP	X1	X2	X3	CD	PS
GRID	100100	1	0.000	0.000	0.000	1	0
GRID	101100	1	.750	0.000	.525	1	0
GRID	101200	1	.750	30.000	.525	1	0
GRID	102100	1	1.500	0.000	1.050	1	0
GRID	102101	1	1.500	15.800	1.050	1	0
GRID	102200	1	1.500	30.000	1.050	1	0

***** PTRIA2 CARDS

PTRIA2	PID	MID	T	NSM
PTRIA2	1	1	.10000	0.00000

*****TRIA2 CARDS

CTRIA2	EID	PID	G1	G2	G3	TH	
CTRIA2	100100	1	101200	101100	100100	0.0	
CTRIA2	101101	1	102101	102100	101100	0.0	
CTRIA2	101102	1	101100	101200	102101	0.0	
CTRIA2	101103	1	102200	102101	101200	0.0	
PLOAD	1	1.0	101200	101100	100100		
PLOAD	1	1.0	102101	102100	101100		
PLOAD	1	1.0	101100	101200	102101		
PLOAD	1	1.0	102200	102101	101200		
GRID	103100	1	2.250	0.000	1.575	1	0
GRID	103101	1	2.250	15.000	1.575	1	0
GRID	103200	1	2.250	30.000	1.575	1	0
GRID	200100	1	3.000	0.000	2.101	1	0
GRID	200101	1	3.000	15.000	2.101	1	0
GRID	200200	1	3.000	30.000	2.101	1	0

*****PQUAD CARDS

PQUAD2	PID	MID	T	NSM
PQUAD2	1	1	.10000	0.00000

*****QUAD CARDS

CQUAD2	EID	PID	G1	G2	G3	G4	TH
CQUAD2	102100	1	102100	102101	103101	103100	0.0

DATA GENERATOR OUTPUT DECK

CQUAD2	102101	1	102101	102200	103200	103101	0.0
CQUAD2	103100	1	103100	103101	200101	200100	0.0
CQUAD2	103101	1	103101	103200	200200	200101	0.0
PLOAD	1	1.0	102100	102101	103101	103100	
PLOAD	1	1.0	102101	102200	103200	103101	
PLOAD	1	1.0	103100	103101	200101	200100	
PLOAD	1	1.0	103101	103200	200200	200101	

```
D A T A   G E N E R A T O R   O U T P U T   D E C K
(11)---- SECTION   7

        DEFINED BY USER
    LENGTH OF SECTION (L)=   6.0000

    AZIMUTHAL ANGLE (THETA)=  -0.0000

    SLOPE OF ARL1 (PHI1)=   90.000

    SLOPE OF ARL2 (PHI2)=   -0.000

         THE GRID POINT NUMBER AT THE INTERSECTION OF Z1,A1( 5, 1) IS     300100

         THE GRID POINT NUMBER AT THE INTERSECTION OF Z2,A1( 7, 1) IS     400100

         THE GRID POINT NUMBER AT THE INTERSECTION OF Z1,A2( 5, 2) IS     300200

         THE GRID POINT NUMBER AT THE INTERSECTION OF Z2,A2( 7, 2) IS     400200


    THE RADIUS AT EACH CORNER... RA=  5.00 RB=   5.00 RC=   5.00 RD=   5.000
```

D A T A   G E N E R A T O R   O U T P U T   D E C K

SECTION BOUNDED BY: ZRPS= 5 7 AND ARPS 1 2

*****GRID CARDS

GRID	ID	CP	X1	X2	X3	CD	PS
GRID	301100	1	5.000	0.000	6.157	1	0
GRID	301101	1	5.000	15.000	6.157	1	0
GRID	301200	1	5.000	30.000	6.157	1	0
GRID	302100	1	5.000	0.000	7.357	1	0
GRID	302101	1	5.000	15.000	7.357	1	0
GRID	302200	1	5.000	30.000	7.357	1	0
GRID	303100	1	5.000	0.000	8.557	1	0
GRID	303101	1	5.000	15.000	8.557	1	0
GRID	303200	1	5.000	30.000	8.557	1	0

*****QUAD CARDS

12

CQUAD2	EID	PID	G1	G2	G3	G4	TH
CQUAD2	300100	1	300100	300101	301101	301100	0.0
CQUAD2	300101	1	300101	300200	301200	301101	0.0
CQUAD2	301100	1	301100	301101	302101	302100	0.0
CQUAD2	301101	1	301101	301200	302200	302101	0.0
CQUAD2	302100	1	302100	302101	303101	303100	0.0
CQUAD2	302101	1	302101	302200	303200	303101	0.0
PLOAD	1	1.0	300100	300101	301101	301100	
PLOAD	1	1.0	300101	300200	301200	301101	
PLOAD	1	1.0	301100	301101	302101	302100	
PLOAD	1	1.0	301101	301200	302200	302101	
PLOAD	1	1.0	302100	302101	303101	303100	
PLOAD	1	1.0	302101	302200	303200	303101	
GRID	304100	1	5.000	0.000	9.757	1	0
GRID	304101	1	5.000	10.000	9.757	1	0
GRID	304102	1	5.000	20.000	9.757	1	0
GRID	304200	1	5.000	30.000	9.757	1	0
GRID	400100	1	5.000	0.000	10.957	1	0
GRID	400101	1	5.000	7.500	10.957	1	0
GRID	400102	1	5.000	15.000	10.957	1	0
GRID	400103	1	5.000	22.500	10.957	1	0
GRID	400200	1	5.000	30.000	10.957	1	0

*****TRIA2 CARDS

CTRIA2	EID	PID	G1	G2	G3	TH
CTRIA2	303100	1	304101	304100	303100	0.0
CTRIA2	303101	1	303100	303101	304101	0.0
CTRIA2	303102	1	304102	304101	303101	0.0

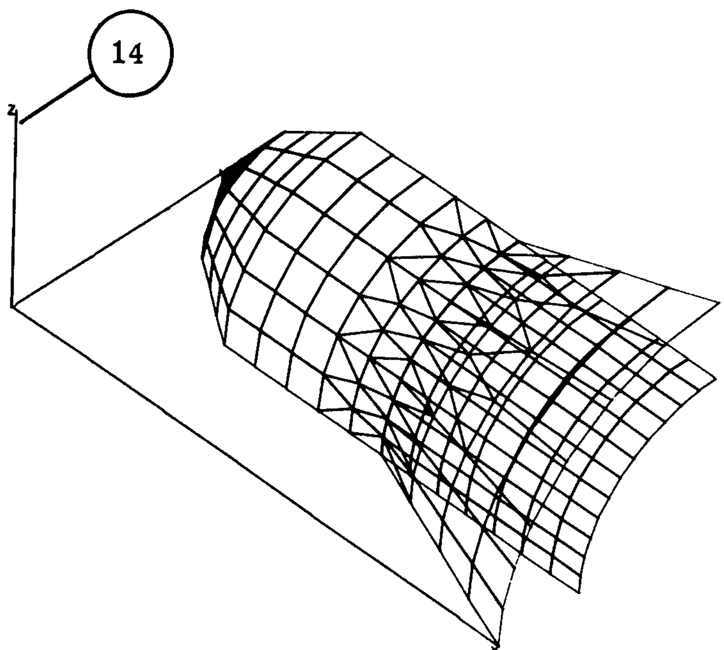
DATA GENERATOR OUTPUT DECK

CTRIA2	303103	1	303101	303200	304102	0.0
CTRIA2	303104	1	304200	304102	303200	0.0
CTRIA2	304100	1	400101	400100	304100	0.0
CTRIA2	304101	1	304100	304101	400101	0.0
CTRIA2	304102	1	400102	400101	304101	0.0
CTRIA2	304103	1	304101	304102	400102	0.0
CTRIA2	304104	1	400103	400102	304102	0.0
CTRIA2	304105	1	304102	304200	400103	0.0
CTRIA2	304106	1	400200	400103	304200	0.0
PLOAD	1	1.0	304101	304100	303100	
PLOAD	1	1.0	303100	303101	304101	
PLOAD	1	1.0	304102	304101	303101	
PLOAD	1	1.0	303101	303200	304102	
PLOAD	1	1.0	304200	304102	303200	
PLOAD	1	1.0	400101	400100	304100	
PLOAD	1	1.0	304100	304101	400101	
PLOAD	1	1.0	400102	400101	304101	
PLOAD	1	1.0	304101	304102	400102	
PLOAD	1	1.0	400103	400102	304102	
PLOAD	1	1.0	304102	304200	400103	
PLOAD	1	1.0	400200	400103	304200	

```
        MESSAGES FROM STRUCTURE PLOTTER

(13)    SUBROUTINE PLOT.     228 ELEMENTS AND     216 GRID POINTS.

        SUBROUTINE MODEL.  PERSPECTIVE PLOTTING AT    .78540 AND    .78540 RADIANS.

        SUBROUTINE PLOT.  ONE PLOT WITH ITYPE =  1 I2D =  2 I3D = -2 IXYZ =  4

        SUBROUTINE PLOT.  ONE PLOT WITH ITYPE =  1 I2D =  2 I3D = -2 IXYZ =  3

        SUBROUTINE PLOT.  ONE PLOT WITH ITYPE =  1 I2D =  2 I3D = -2 IXYZ =  2

END OF PLOTTING.  NO. OF FRAMES     4


*******   701040  END OF APPLICATION  *******
```

DEMONSTRATION CONE...   MCKEE 02/25/75 5

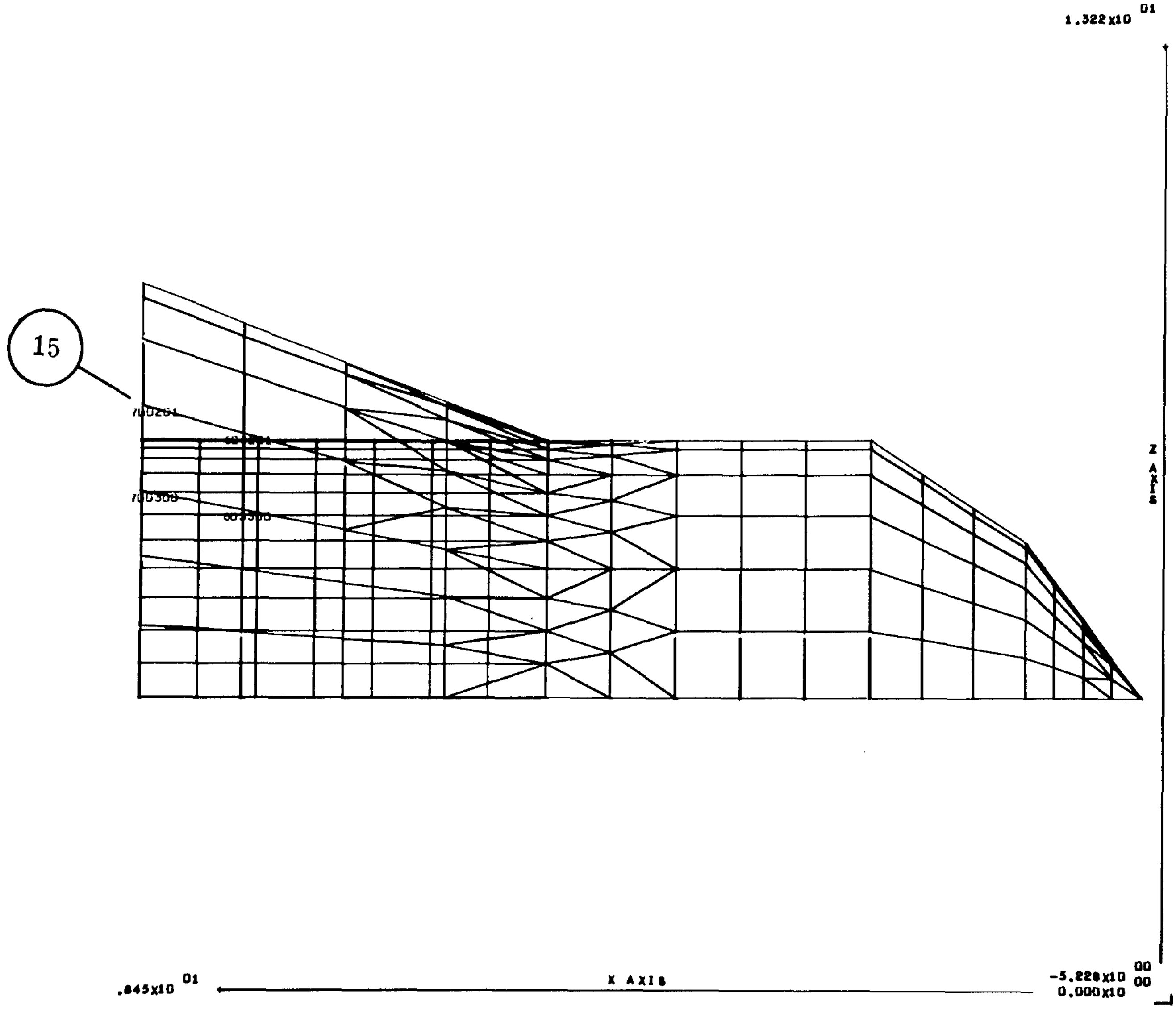
PERSPECTIVE PLOT



TWO DIMENSIONAL ELEMENTS

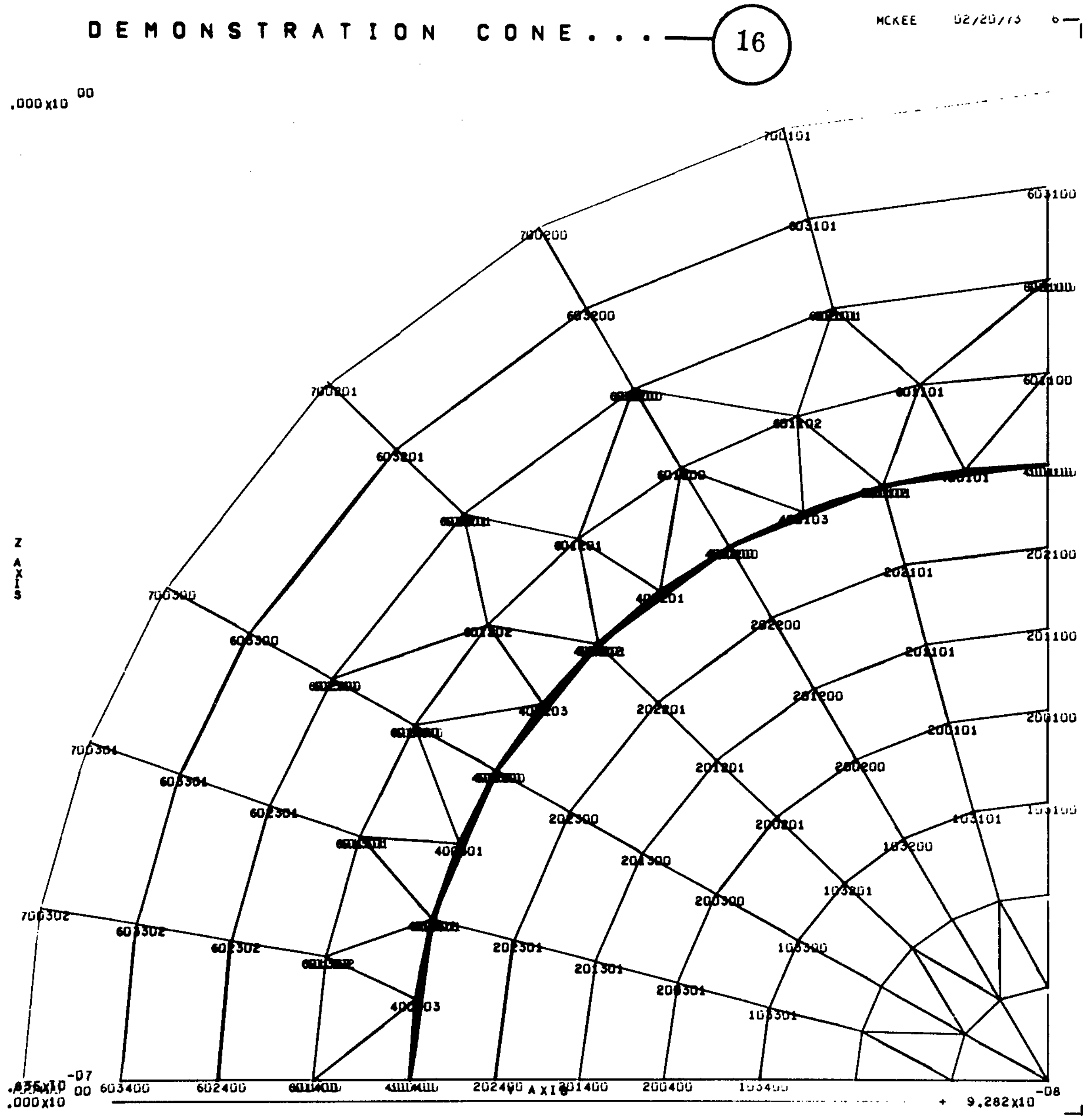
DEMONSTRATION CONE...

MCKEE 02/20/73 5



DEMONSTRATION CONE...

16



4.2 POINT LOAD ON CYLINDER

The purpose of this data generator application was to generate an idealization of a ring-stiffened cylinder which could be analyzed with a concentrated load applied at a point on one stiffener. The symmetry of the structure and the load required only half of the cylinder to be modelled. A listing of the bulk data deck and three structural plots have been included.

CARD COUNT INPUT DATA DECK

1	CONE	1	1	2	2	90.		4.	15.+C001
2	+C001	53.744	.512	1.	8	1			+C002
3	+C002				6				+C003
4	+C003				7				+C004
5	+C004				5				
6	CONE	2	1	3	2	90.		15.555	+C005
7	+C005	53.744	.512	1.		1			+C006
8	+C006				5				
9	CONE	3	1	4	2	90.		20.55	+C007
10	+C007	53.744	.512	1.	3	1			+C008
11	+C008				4				
12	CONE	4	1	5	2	90.		20.55	+C009
13	+C009	53.744	.512	1.	2	1			+C010
14	+C010				4				
15	CONE	1	2	2	3				75.+C011
16	+C011	53.744	.512	1.	9	1			
17	CONE	2	2	3	3				+C012
18	+C012	53.744	.512	1.		1			+C013
19	+C013				5				
20	CONE	3	2	4	3				+C014
21	+C014	53.744	.512	1.	7	1			
22	CONE	4	2	5	3				+C015
23	+C015	53.744	.512	1.	6	1			
24	CONE	1	3	2	4				90.+C016
25	+C016	53.744	.512	1.	6	1			+C017
26	+C017				2				+C018
27	+C018								+C019
28	+C019				1				
29	CONE	2	3	3	4				+C020
30	+C020	53.744	.512	1.		1			+C021
31	+C021				3				
32	CONE	3	3	4	4				+C022
33	+C022	53.744	.512	1.		1			
34	CONE	4	3	5	4				+C023
35	+C023	53.744	.512	1.		1			
36	ZEQU	6	1	8	3	9	4	10	5+100
37	+100	15	14	19	18				
38	CONE	6	1	11	2				+101
39	+101		.512						+102
40	+102	48.744			3				
41	CONE	11	1	12	2	90.		2.5	+103
42	+103		.512		6				+104
43	+104				2				
44	CONE	8	1	14	2				+105
45	+105		.512						+106
46	+106	48.744			3				
47	CONE	14	1	13	2	-90.		2.5	+107
48	+107		.512						+108
49	+108				2				
50	CONE	15	1	16	2	90		2.5	+109

CARD COUNT   I N P U T   D A T A   D E C K

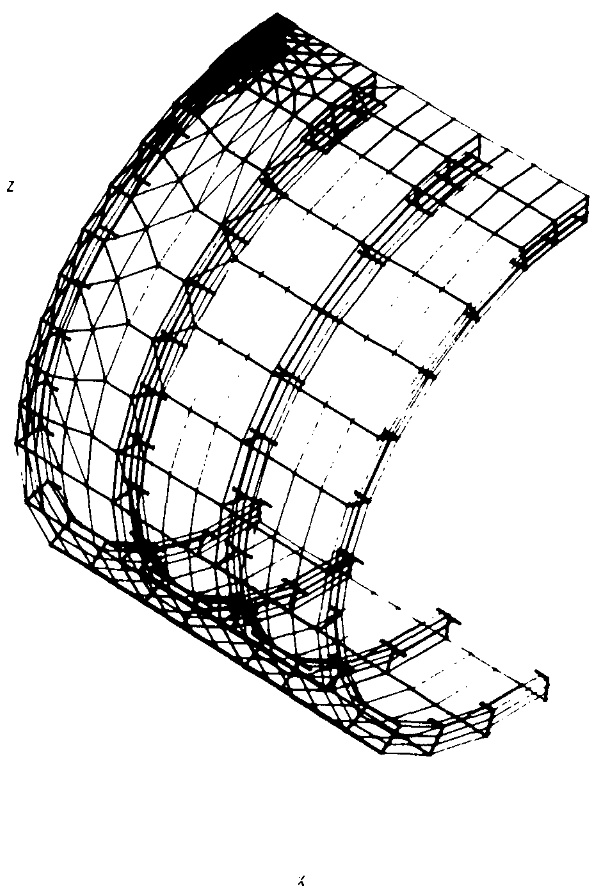
51	+109		.512					+110
52	+110				2			
53	CONE	9	1	18	2			+111
54	+111		.512					+112
55	+112	48.744			2			
56	CONE	18	1	17	2	-90	2.5	+113
57	+113		.512					+114
58	+114				2			
59	CONE	19	1	20	2	90.	2.5	+115
60	+115		.512					+116
61	+116				2			
62	CONE	10	1	22	2			+117
63	+117		.512					+118
64	+118	48.744			3			
65	CONE	22	1	21	2	-90	2.5	+119
66	+119		.512					+120
67	+120				2			
68	CONE	6	2	11	3			+121
69	+121		.512					
70	CONE	11	2	12	3		2.5	+123
71	+123		.512		7			
72	CONE	8	2	14	3			+124
73	+124		.512					
74	CONE	14	2	13	3		2.5	+126
75	+126		.512					
76	CONE	15	2	16	3		2.5	+127
77	+127		.512					
78	CONE	9	2	18	3			+128
79	+128		.512					
80	CONE	18	2	17	3		2.5	+130
81	+130		.512					
82	CONE	19	2	20	3		2.5	+131
83	+131		.512					
84	CONE	10	2	22	3			+132
85	+132		.512					
86	CONE	22	2	21	3		2.5	+133
87	+133		.512					
88	CONE	6	3	11	4			+134
89	+134		.512					
90	CONE	11	3	12	4		2.5	+135
91	+135		.512					
92	CONE	8	3	14	4			+136
93	+136		.512					
94	CONE	14	3	13	4		2.5	+137
95	+137		.512					
96	CONE	15	3	16	4		2.5	+138
97	+138		.512					
98	CONE	9	3	18	4			+139
99	+139		.512					
100	CONE	18	3	17	4		2.5	+140

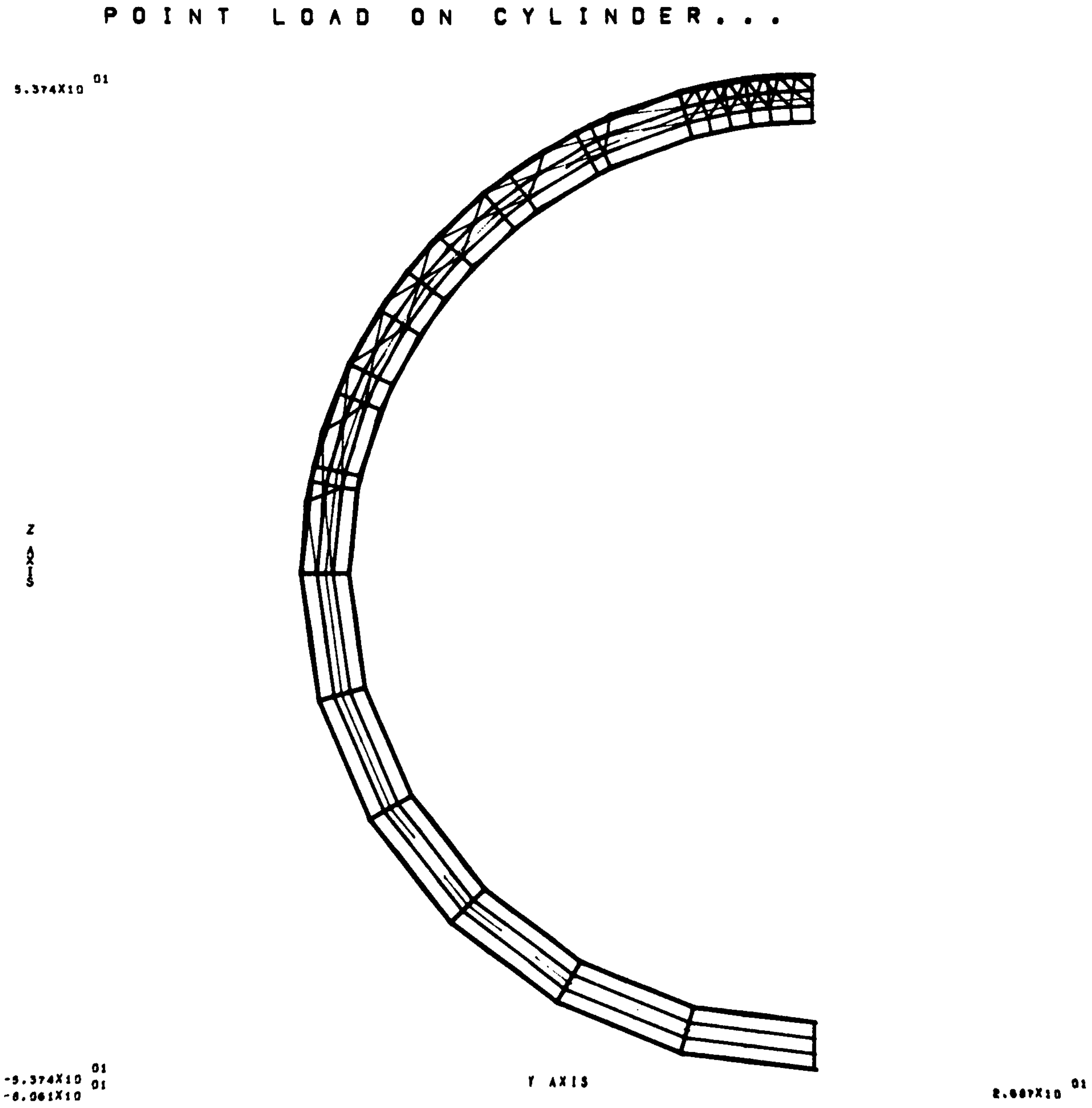
CARD COUNT INPUT DATA DECK

101	+140		.512				
102	CONE	19	3	20	4	2.5	+141
103	+141		.512				
104	CONE	10	3	22	4		+142
105	+142		.512				
106	CONE	22	3	21	4	2.5	+143
107	+143		.512				
108	ENDDATA						

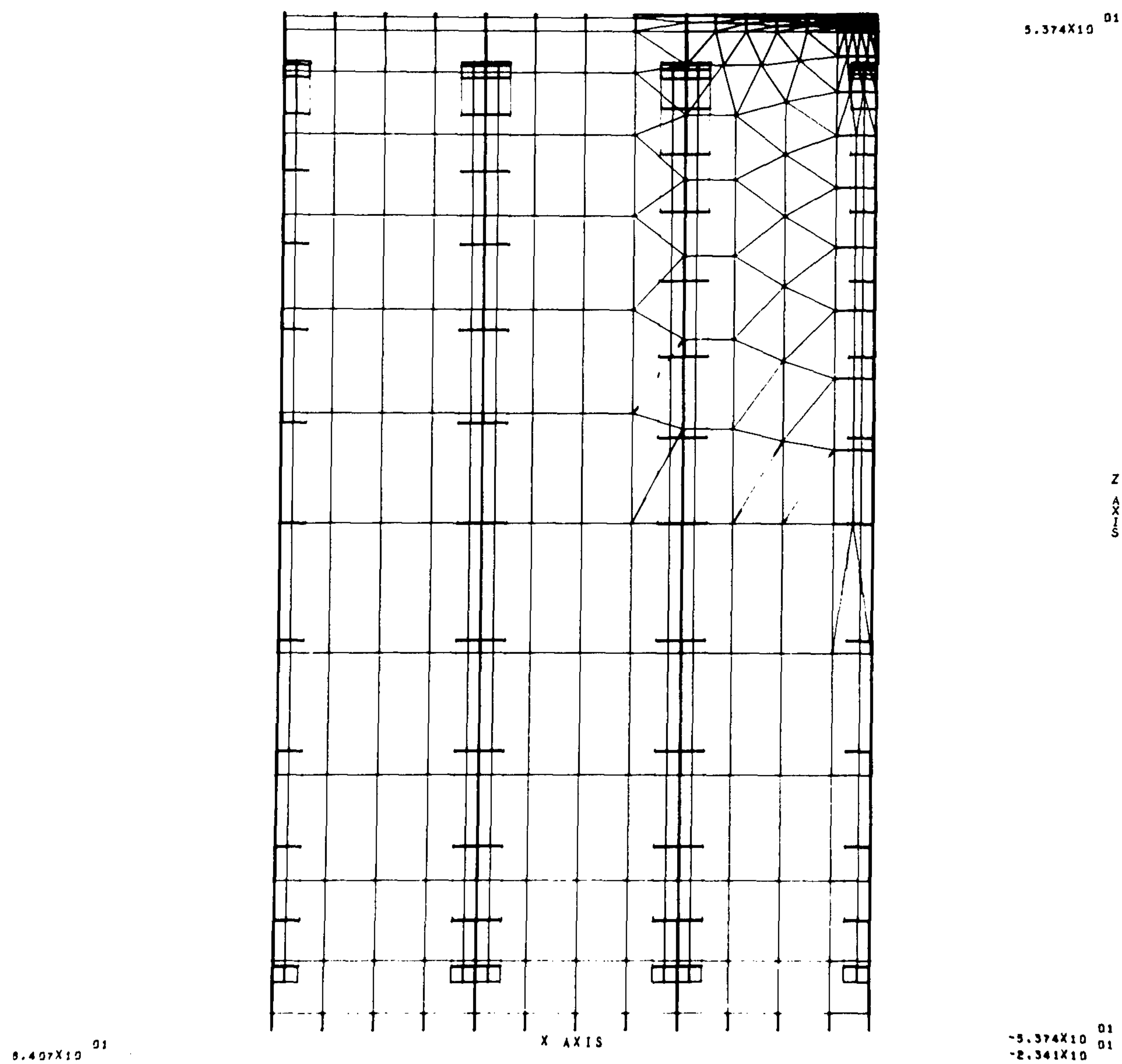
POINT LOAD ON CYLINDER...

PERSPECTIVE PLOT




POINT LOAD ON CYLINDER...
5.374X10 01
Z AXIS
-5.374X10 01
-6.061X10 01
Y AXIS
2.687X10 01

POINT LOAD ON CYLINDER...



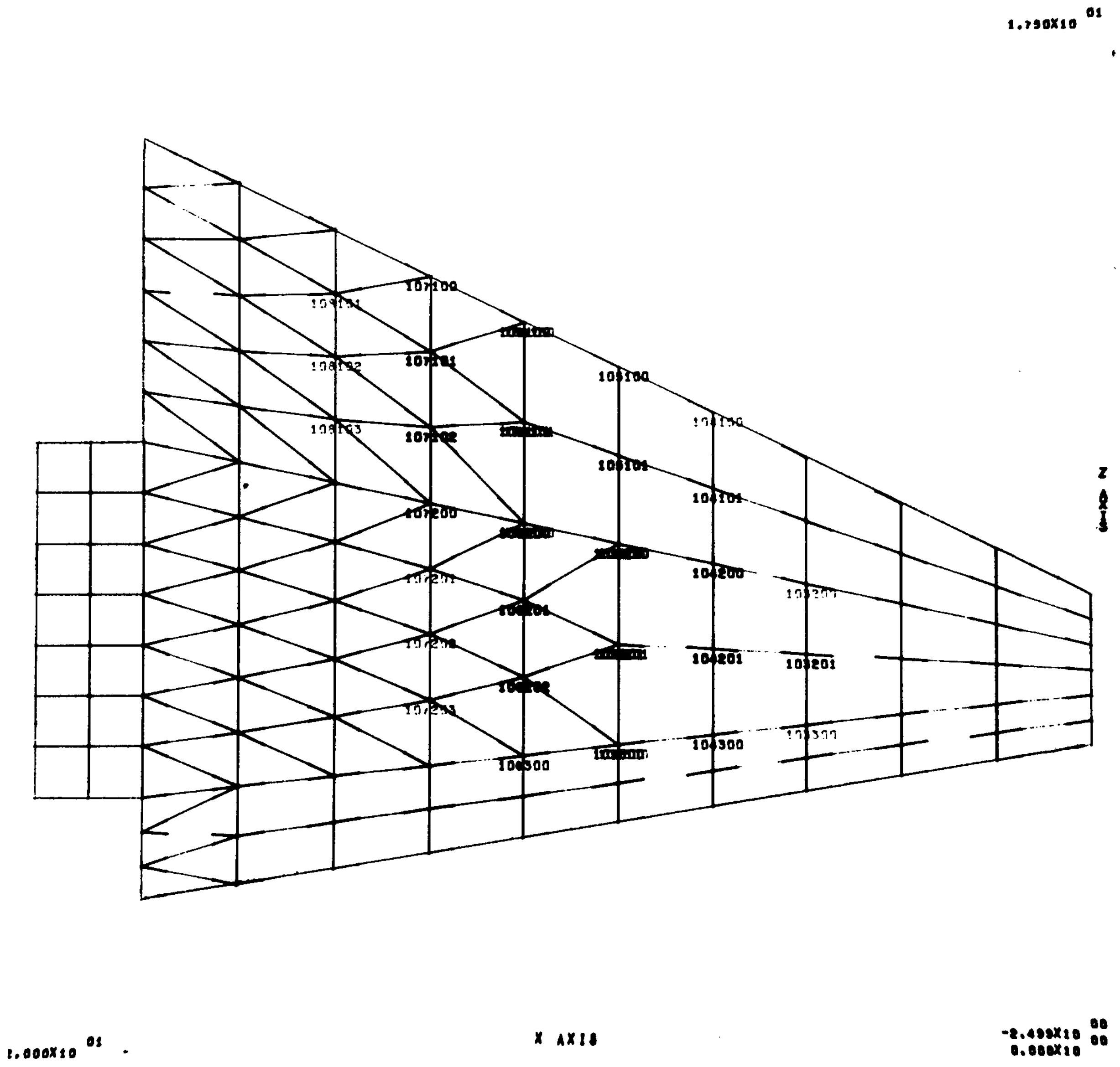
4.3 PLANFORM STABILIZER

This is an example of the use of the data generator to idealize planer structures. A listing of the user supplied data and a structural plot have been included. Please note that the model has been generated in the r-z-plane and that it was generated from right to left as seen in the plot.

```
                                                                                        03/01/73          PAGE      1
    CARD COUNT              E X E C U T I O N   C O N T R O L   D E C K

        1        ID MCKEE, STABILIZER
        2       $TITLE = P L A N F O R M   S T A B I L I Z E R
        3       $PLOTID = MCKEE, CODE1844, EXT71493
        4       $PUNCH = YES
        5        BEGIN BULK
```

```
P L A N F O R M   S T A B I L I Z E R
                                                                                        03/01/73          PAGE      2
    CARD COUNT                  I N P U T   D A T A   D E C K

        1       CONE          1      10       2      20
        2       +1          6.0     .25      1.       2                   18.0       +1
        3       +2          15.      .5              10                              +2
        4       +3          9.0     .75               6                              +3
        5       +4          5.0                                                      +4
        6       CONE          1      20       2      30
        7       +5                  .25      2.       2                              +5
        8       +6                  .75                                              +6
        9       +7          2.0                                                      +7
       10       +8          4.0                       7                              +8
       11       CONE          1      30       2      40
       12       +9                  .25      1.       2                              +9
       13       +10                 .75                                              +10
       14       +11        .001     .25                                              +11
       15       +12         3.0                       3                              +12
       16       CONE          2      20       3      30
       17       +13                 .75                                2.0       +13
       18       +14         9.0                                                      +14
       19       +15         2.0                       2                              +15
       20       ENDDATA
```

PLANFORM STABILIZER



4.4 AXISYMMETRIC SUBMARINE

This example illustrates the data setup required to follow the data generation run with a pass through the BANDIT program for bandwidth reduction and then with a frequency response analysis using NASTRAN. A listing of the user-supplied data, structural plots, and a listing of the generated data have been included.

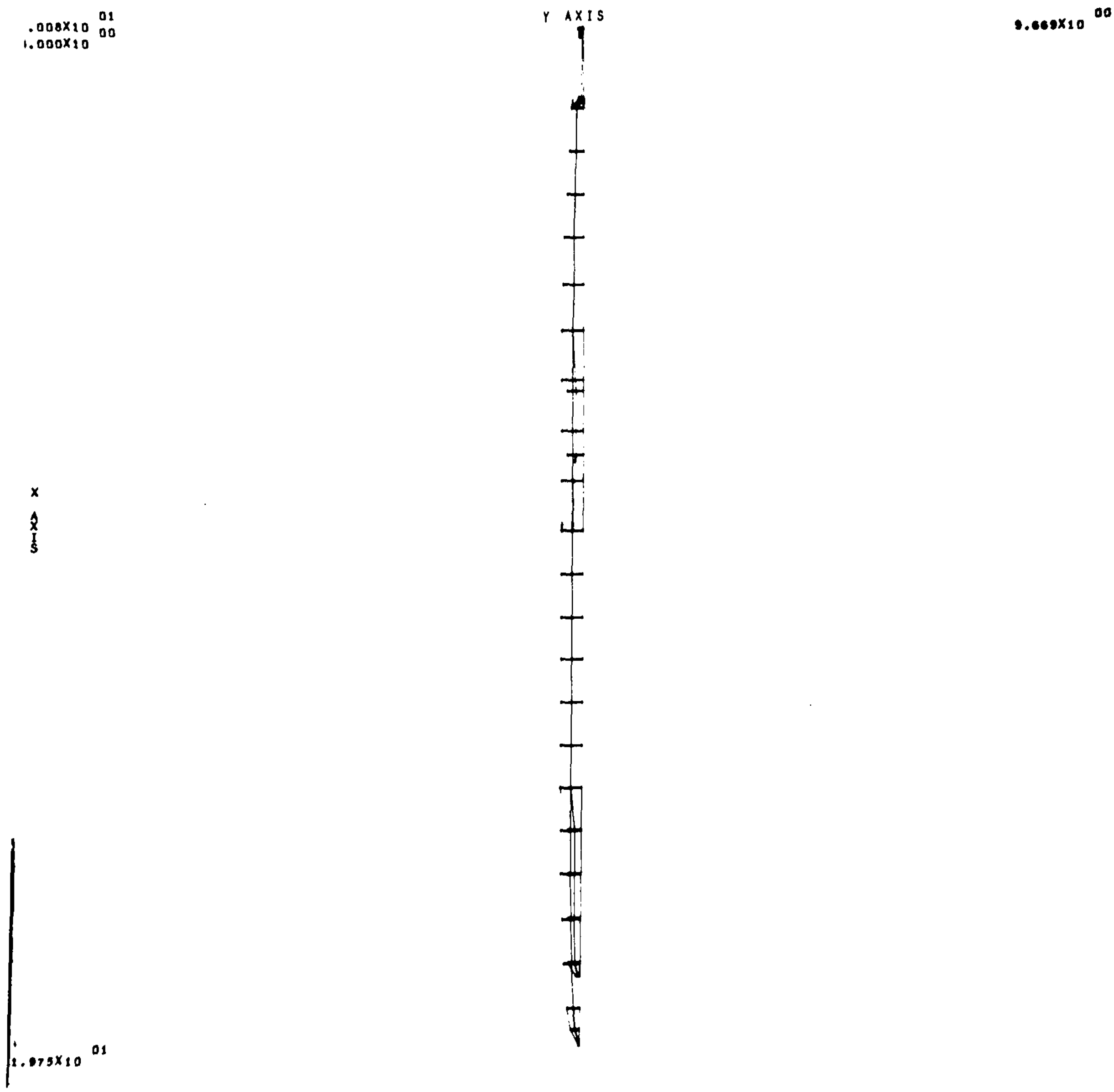
```
CARD COUNT              E X E C U T I O N   C O N T R O L   D E C K

    1       ID MCKEE,  SUBMARINE
    2      $NASTRAN = YES
    3      $TITLE= A X I S Y M M E T R I C   S U B M A R I N E   ( 2 2 . 5   D E G )
    4      $PLOTID = MCKEE, CODE1844, EXT71493
    5      $PUNCH = YES
    6       $END
    7       APP DISP
    8       SOL 8,0
    9       TIME 9
   10       DIAG 13
   11       CEND
   12       SPC = 1
   13       ECHO = NONE
   14       DLOAD = 100
   15       FREQUENCY = 1000
   16       TITLE= A X I S Y M M E T R I C   S U B M A R I N E   ( 2 2 . 5   D E G )
   17       SUBTITLE = 0.1 THICKNESS
   18       $SEQUENCE YES
   19       $PUNCH NONE
   20       $PRINT MAX
   21       BEGIN BULK
```

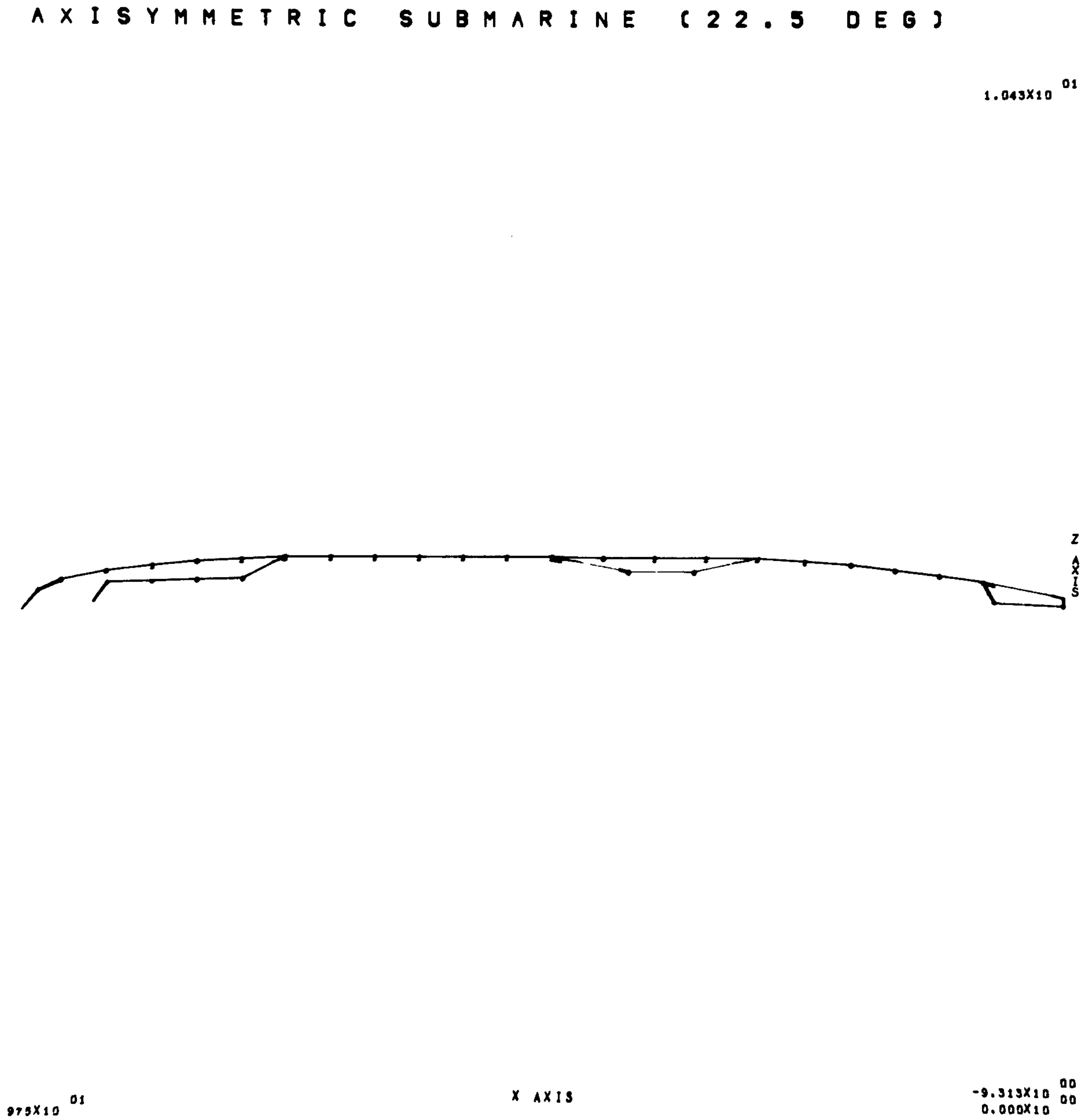
CARD COUNT	INPUT DATA DECK							
1	CONE	1	1	2	2	0.0	22.5	+011
2	+011	.1	.01		2			+012
3	+012	.27			1			
4	CONE	2	1	3	2	1.5		+021
5	+021		.01					+022
6	+022	.6			1			
7	CONE	3	1	5	2	2.5		+031
8	+031		.01					+032
9	+032	.94			3			
10	CONE	5	1	6	2	1.8		+041
11	+041		.01					+042
12	+042	1.05			2			
13	CONE	6	1	11	2	3.94		+051
14	+051		.01					+052
15	+052	1.076			4			
16	CONE	11	1	12	2	5.07		+061
17	+061		.01					+062
18	+062	1.076			6			
19	CONE	12	1	13	2	1.66		+071
20	+071		.01					+072
21	+072	1.005			2			
22	CONE	13	1	14	2	1.71		+081
23	+081		.01					+082
24	+082	.805			2			
25	CONE	14	1	15	2	.85		+091
26	+091		.01					+092
27	+092	.627			1			
28	CONE	15	1	16	2	.42		+101
29	+101		.01					+102
30	+102	.414			1			
31	CONE	16	1	17	2	.30		+111
32	+111		.01					+112
33	+112	.05			1			
34	CONE	23	1	4	2	1.3		+121
35	+121		.01					+122
36	+122	.2			1			
37	CONE	4	1	3	2	.2		+131
38	+131		.01					+132
39	+132				2			
40	CONE	21	1	8	2	1.2		+141
41	+141		.01					+142
42	+142	.78			1			
43	CONE	8	1	10	2	1.25		+151
44	+151		.01					+152
45	+152	.78			1			
46	CONE	10	1	11	2	1.49		+161
47	+161		.01					+162
48	+162	1.076			1			
49	CONE	22	1	18	2	.81		+171
50	+171		.01					+172

CARD COUNT   I N P U T   D A T A   D E C K

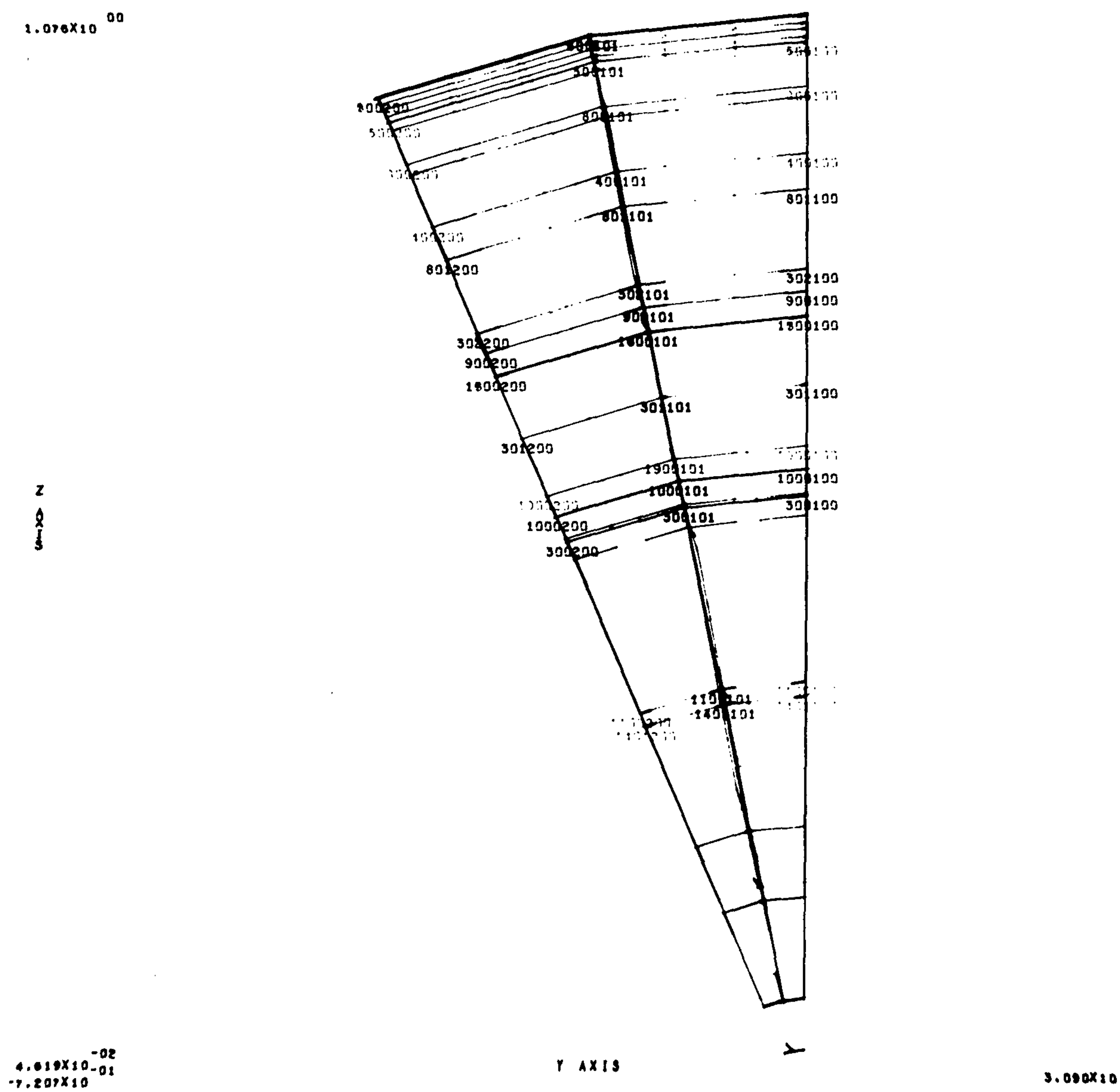
51	+172	.65			1					
52	CONE	18	1	19	2			2.55		+181
53	+181		.01							+182
54	+182	.58			3					
55	CONE	19	1	20	2			.24		+191
56	+191		.01							+192
57	+192	.214			1					
58	ZEQU	21	6	22	12	23	1			
59	$END									
60	DAREA	1	1200100	3	.333333	1200101	3	.333333		
61	DAREA	1	1200200	3	.333333					
62	FREQ1	1000	0.0	50.0	6					
63	MAT1	1	3.E7		.3	1.0				
64	PARAM	DECOMOPT	4							
65	SPC1	1	246	100100	200100	300100	301100	302100	400100	
66	SPC1	1	246	401100	500100	501100	502100	503100	600100	
67	SPC1	1	246	601100	602100	603100	604100	605100	700100	
68	SPC1	1	246	800100	801100	900100	1000100	1100100	1200100	
69	SPC1	1	246	1400100	1401100	1600100	1700100	1900100	1901100	
70	SPC1	1	246	1902100	2000100	2100100				
71	SPC1	1	246	701100						
72	SPC1	1	246	100200	200200	300200	301200	302200	400200	
73	SPC1	1	246	401200	500200	501200	502200	503200	600200	
74	SPC1	1	246	601200	602200	603200	604200	605200	700200	
75	SPC1	1	246	701200	800200	801200	900200	1000200	1100200	
76	SPC1	1	246	1200200	1400200	1401200	1600200	1700200	1900200	
77	SPC1	1	246	1901200	1902200	2000200	2100200			
78	RLOAD2	100	1			10				
79	TABLED1	10								+T10
80	+T10	0.0	1.0	1000.0	1.0	ENDT				
81	ENDDATA									

AXISYMMETRIC SUBMARINE (22.5 DEG)




AXISYMMETRIC SUBMARINE (22.5 DEG)
1.043X10 01
Z AXIS
975X10 01
X AXIS
-9.313X10 00
0.000X10 00

AXISYMMETRIC SUBMARINE (22.5 DEG)



```
ID MCKEE,  SUBMARINE
$   NASTRAN = YES
$   TITLE= A X I S Y M M E T R I C   S U B M A R I N E  ( 2 2 . 5   D E G )
$   PLOTID = MCKEE, CODE1844, EXT71493
$   PUNCH = YES
$END
APP DISP
SOL 8,0
TIME 9
DIAG 13
CEND
SPC = 1
ECHO = NONE
DLOAD = 100
FREQUENCY = 1000
TITLE= A X I S Y M M E T R I C   S U B M A R I N E  ( 2 2 . 5   D E G )
SUBTITLE = 0.1 THICKNESS
$SEQUENCE YES
$PUNCH NONE
$PRINT MAX
BEGIN BULK
DAREA          1 1200100       3 .333333 1200101       3 .333333
DAREA          1 1200200       3 .333333
FREQ1       1000     0.0    50.0       6
MAT1           1    3.E7              .3     1.0
PARAM   DECOMOPT       4
SPC1           1     246  100100  200100  300100  301100  302100  400100
SPC1           1     246  401100  500100  501100  502100  503100  600100
SPC1           1     246  601100  602100  603100  604100  605100  700100
SPC1           1     246  800100  801100  900100 1000100 1100100 1200100
SPC1           1     246 1400100 1401100 1600100 1700100 1900100 1901100
SPC1           1     246 1902100 2000100 2100100
SPC1           1     246  701100
SPC1           1     246  100200  200200  300200  301200  302200  400200
SPC1           1     246  401200  500200  501200  502200  503200  600200
SPC1           1     246  601200  602200  603200  604200  605200  700200
SPC1           1     246  701200  800200  801200  900200 1000200 1100200
SPC1           1     246 1200200 1400200 1401200 1600200 1700200 1900200
SPC1           1     246 1901200 1902200 2000200 2100200
RLOAD2       100       1                      10
TABLED1       10                                                        +T10
+T10         0.0     1.0  1000.0     1.0    ENDT
CQUAD2    100100       1  100100  100101  200101  200100     0.0
CQUAD2    100101       1  100101  100200  200200  200101     0.0
CQUAD2    200100       1  200100  200101  300101  300100     0.0
CQUAD2    200101       1  200101  200200  300200  300101     0.0
CQUAD2    300100       1  300100  300101  301101  301100     0.0
CQUAD2    301100       1  301100  301101  302101  302100     0.0
CQUAD2    302100       1  302100  302101  400101  400100     0.0
CQUAD2    300101       1  300101  300200  301200  301101     0.0
CQUAD2    301101       1  301101  301200  302200  302101     0.0
CQUAD2    302101       1  302101  302200  400200  400101     0.0
CQUAD2    400100       1  400100  400101  401101  401100     0.0
CQUAD2    401100       1  401100  401101  500101  500100     0.0
CQUAD2    400101       1  400101  400200  401200  401101     0.0
CQUAD2    401101       1  401101  401200  500200  500101     0.0
CQUAD2    500100       1  500100  500101  501101  501100     0.0
CQUAD2    501100       1  501100  501101  502101  502100     0.0
CQUAD2    502100       1  502100  502101  503101  503100     0.0
CQUAD2    503100       1  503100  503101  600101  600100     0.0
CQUAD2    500101       1  500101  500200  501200  501101     0.0
CQUAD2    501101       1  501101  501200  502200  502101     0.0
```

```
CQUAD2    502101     1  502101  502200  503200  503101    0.0
CQUAD2    503101     1  503101  503200  600200  600101    0.0
CQUAD2    600100     1  600100  600101  601101  601100    0.0
CQUAD2    601100     1  601100  601101  602101  602100    0.0
CQUAD2    602100     1  602100  602101  603101  603100    0.0
CQUAD2    603100     1  603100  603101  604101  604100    0.0
CQUAD2    604100     1  604100  604101  605101  605100    0.0
CQUAD2    605100     1  605100  605101  700101  700100    0.0
CQUAD2    600101     1  600101  600200  601200  601101    0.0
CQUAD2    601101     1  601101  601200  602200  602101    0.0
CQUAD2    602101     1  602101  602200  603200  603101    0.0
CQUAD2    603101     1  603101  603200  604200  604101    0.0
CQUAD2    604101     1  604101  604200  605200  605101    0.0
CQUAD2    605101     1  605101  605200  700200  700101    0.0
CQUAD2    700100     1  700100  700101  701101  701100    0.0
CQUAD2    701100     1  701100  701101  800101  800100    0.0
CQUAD2    700101     1  700101  700200  701200  701101    0.0
CQUAD2    701101     1  701101  701200  800200  800101    0.0
CQUAD2    800100     1  800100  800101  801101  801100    0.0
CQUAD2    801100     1  801100  801101  900101  900100    0.0
CQUAD2    800101     1  800101  800200  801200  801101    0.0
CQUAD2    801101     1  801101  801200  900200  900101    0.0
CQUAD2    900100     1  900100  900101 1000101 1000100    0.0
CQUAD2    900101     1  900101  900200 1000200 1000101    0.0
CQUAD2   1000100     1 1000100 1000101 1100101 1100100    0.0
CQUAD2   1000101     1 1000101 1000200 1100200 1100101    0.0
CQUAD2   1100100     1 1100100 1100101 1200101 1200100    0.0
CQUAD2   1100101     1 1100101 1100200 1200200 1200101    0.0
CQUAD2   1300100     1  100100  100101 1400101 1400100    0.0
CQUAD2   1300101     1  100101  100200 1400200 1400101    0.0
CQUAD2   1400100     1 1400100 1400101 1401101 1401100    0.0
CQUAD2   1401100     1 1401100 1401101  300101  300100    0.0
CQUAD2   1400101     1 1400101 1400200 1401200 1401101    0.0
CQUAD2   1401101     1 1401101 1401200  300200  300101    0.0
CQUAD2   1500100     1  500100  500101 1600101 1600100    0.0
CQUAD2   1500101     1  500101  500200 1600200 1600101    0.0
CQUAD2   1600100     1 1600100 1600101 1700101 1700100    0.0
CQUAD2   1600101     1 1600101 1600200 1700200 1700101    0.0
CQUAD2   1700100     1 1700100 1700101  600101  600100    0.0
CQUAD2   1700101     1 1700101 1700200  600200  600101    0.0
CQUAD2   1800100     1  700100  700101 1900101 1900100    0.0
CQUAD2   1800101     1  700101  700200 1900200 1900101    0.0
CQUAD2   1900100     1 1900100 1900101 1901101 1901100    0.0
CQUAD2   1901100     1 1901100 1901101 1902101 1902100    0.0
CQUAD2   1902100     1 1902100 1902101 2000101 2000100    0.0
CQUAD2   1900101     1 1900101 1900200 1901200 1901101    0.0
CQUAD2   1901101     1 1901101 1901200 1902200 1902101    0.0
CQUAD2   1902101     1 1902101 1902200 2000200 2000101    0.0
CQUAD2   2000100     1 2000100 2000101 2100101 2100100    0.0
CQUAD2   2000101     1 2000101 2000200 2100200 2100101    0.0
CORD2C         1           0.0     0.0     0.0     1.0    0.0       0.0+CORD2C1
+CORD2C1     0.0     0.0   1.0
$    SECTION BOUNDED BY ZRLS     1     2 AND  ARLS      1      2
GRID      100100     1    .100   0.000   0.000       1      0
GRID      200100     1    .270   0.000   0.000       1      0
GRID      100101     1    .100  11.250   0.000       1      0
GRID      200101     1    .270  11.250   0.000       1      0
GRID      100200     1    .100  22.500   0.000       1      0
GRID      200200     1    .270  22.500   0.000       1      0
$    SECTION BOUNDED BY ZRLS     2     3 AND  ARLS      1      2
GRID      300100     1    .600   0.000   1.500       1      0
GRID      300101     1    .600  11.250   1.500       1      0
GRID      300200     1    .600  22.500   1.500       1      0
$    SECTION BOUNDED BY ZRLS     3     5 AND  ARLS      1      2
GRID      301100     1    .713   0.000   2.333       1      0
GRID      302100     1    .827   0.000   3.167       1      0
```

```
GRID     400100    1    .940   0.000   4.000      1      0
GRID     301101    1    .713  11.250   2.333      1      0
GRID     302101    1    .827  11.250   3.167      1      0
GRID     400101    1    .940  11.250   4.000      1      0
GRID     301200    1    .713  22.500   2.333      1      0
GRID     302200    1    .827  22.500   3.167      1      0
GRID     400200    1    .940  22.500   4.000      1      0
$   SECTION BOUNDED BY ZRLS    5     6 AND ARLS     1    2
GRID     401100    1    .995   0.000   4.900      1      0
GRID     500100    1   1.050   0.000   5.800      1      0
GRID     401101    1    .995  11.250   4.900      1      0
GRID     500101    1   1.050  11.250   5.800      1      0
GRID     401200    1    .995  22.500   4.900      1      0
GRID     500200    1   1.050  22.500   5.800      1      0
$   SECTION BOUNDED BY ZRLS    6    11 AND ARLS     1    2
GRID     501100    1   1.056   0.000   6.785      1      0
GRID     502100    1   1.063   0.000   7.770      1      0
GRID     503100    1   1.069   0.000   8.755      1      0
GRID     600100    1   1.076   0.000   9.740      1      0
GRID     501101    1   1.056  11.250   6.785      1      0
GRID     502101    1   1.063  11.250   7.770      1      0
GRID     503101    1   1.069  11.250   8.755      1      0
GRID     600101    1   1.076  11.250   9.740      1      0
GRID     501200    1   1.056  22.500   6.785      1      0
GRID     502200    1   1.063  22.500   7.770      1      0
GRID     503200    1   1.069  22.500   8.755      1      0
GRID     600200    1   1.076  22.500   9.740      1      0
$   SECTION BOUNDED BY ZRLS   11    12 AND ARLS     1    2
GRID     601100    1   1.076   0.000  10.585      1      0
GRID     602100    1   1.076   0.000  11.430      1      0
GRID     603100    1   1.076   0.000  12.275      1      0
GRID     604100    1   1.076   0.000  13.120      1      0
GRID     605100    1   1.076   0.000  13.965      1      0
GRID     700100    1   1.076   0.000  14.810      1      0
GRID     601101    1   1.076  11.250  10.585      1      0
GRID     602101    1   1.076  11.250  11.430      1      0
GRID     603101    1   1.076  11.250  12.275      1      0
GRID     604101    1   1.076  11.250  13.120      1      0
GRID     605101    1   1.076  11.250  13.965      1      0
GRID     700101    1   1.076  11.250  14.810      1      0
GRID     601200    1   1.076  22.500  10.585      1      0
GRID     602200    1   1.076  22.500  11.430      1      0
GRID     603200    1   1.076  22.500  12.275      1      0
GRID     604200    1   1.076  22.500  13.120      1      0
GRID     605200    1   1.076  22.500  13.965      1      0
GRID     700200    1   1.076  22.500  14.810      1      0
$   SECTION BOUNDED BY ZRLS   12    13 AND ARLS     1    2
GRID     701100    1   1.041   0.000  15.640      1      0
GRID     800100    1   1.005   0.000  16.470      1      0
GRID     701101    1   1.041  11.250  15.640      1      0
GRID     800101    1   1.005  11.250  16.470      1      0
GRID     701200    1   1.041  22.500  15.640      1      0
GRID     800200    1   1.005  22.500  16.470      1      0
$   SECTION BOUNDED BY ZRLS   13    14 AND ARLS     1    2
GRID     801100    1    .905   0.000  17.325      1      0
GRID     900100    1    .805   0.000  18.180      1      0
GRID     801101    1    .905  11.250  17.325      1      0
GRID     900101    1    .805  11.250  18.180      1      0
GRID     801200    1    .905  22.500  17.325      1      0
GRID     900200    1    .805  22.500  18.180      1      0
$   SECTION BOUNDED BY ZRLS   14    15 AND ARLS     1    2
GRID    1000100    1    .627   0.000  19.030      1      0
GRID    1000101    1    .627  11.250  19.030      1      0
GRID    1000200    1    .627  22.500  19.030      1      0
$   SECTION BOUNDED BY ZRLS   15    16 AND ARLS     1    2
GRID    1100100    1    .414   0.000  19.450      1      0
```

```
GRID     1100101       1     .414   11.250   19.450        1        0
GRID     1100200       1     .414   22.500   19.450        1        0
$   SECTION BOUNDED BY ZRLS     16     17 AND ARLS      1     2
GRID     1200100       1     .050    0.000   19.750        1        0
GRID     1200101       1     .050   11.250   19.750        1        0
GRID     1200200       1     .050   22.500   19.750        1        0
$   SECTION BOUNDED BY ZRLS     23      4 AND ARLS      1     2
GRID     1400100       1     .200    0.000    1.300        1        0
GRID     1400101       1     .200   11.250    1.300        1        0
GRID     1400200       1     .200   22.500    1.300        1        0
$   SECTION BOUNDED BY ZRLS      4      3 AND ARLS      1     2
GRID     1401100       1     .400    0.000    1.400        1        0
GRID     1401101       1     .400   11.250    1.400        1        0
GRID     1401200       1     .400   22.500    1.400        1        0
$   SECTION BOUNDED BY ZRLS     21      8 AND ARLS      1     2
GRID     1600100       1     .780    0.000    7.000        1        0
GRID     1600101       1     .780   11.250    7.000        1        0
GRID     1600200       1     .780   22.500    7.000        1        0
$   SECTION BOUNDED BY ZRLS      8     10 AND ARLS      1     2
GRID     1700100       1     .780    0.000    8.250        1        0
GRID     1700101       1     .780   11.250    8.250        1        0
GRID     1700200       1     .780   22.500    8.250        1        0
$   SECTION BOUNDED BY ZRLS     10     11 AND ARLS      1     2
$   SECTION BOUNDED BY ZRLS     22     18 AND ARLS      1     2
GRID     1900100       1     .650    0.000   15.620        1        0
GRID     1900101       1     .650   11.250   15.620        1        0
GRID     1900200       1     .650   22.500   15.620        1        0
$   SECTION BOUNDED BY ZRLS     18     19 AND ARLS      1     2
GRID     1901100       1     .627    0.000   16.470        1        0
GRID     1902100       1     .603    0.000   17.320        1        0
GRID     2000100       1     .580    0.000   18.170        1        0
GRID     1901101       1     .627   11.250   16.470        1        0
GRID     1902101       1     .603   11.250   17.320        1        0
GRID     2000101       1     .580   11.250   18.170        1        0
GRID     1901200       1     .627   22.500   16.470        1        0
GRID     1902200       1     .603   22.500   17.320        1        0
GRID     2000200       1     .580   22.500   18.170        1        0
$   SECTION BOUNDED BY ZRLS     19     20 AND ARLS      1     2
GRID     2100100       1     .214    0.000   18.410        1        0
GRID     2100101       1     .214   11.250   18.410        1        0
GRID     2100200       1     .214   22.500   18.410        1        0
PQUAD2         1       1  .01000 0.00000
ENDDATA
```

4.5 TEST MISSILE

This data generator application illustrates the techniques required to model a closed 360-degree structure. The fins of this structure are examples of non-symmetric components which can be generated using the CONE module. Note that the user was required to break the circular equivalence specifications generated by the CONEND modules by setting the flag on bulk data card 12. Note also that an unacceptable quadrilateral element (a triangle) was generated at the root of each fin on the leading edge. Only the triangular elements need to be manually replaced since duplicate gridpoints were avoided by using IEQU specifications. A listing of the data deck and two plots have been included.

```
CARD COUNT            E X E C U T I O N   C O N T R O L   D E C K

    1        ID MCKEE,MISSILE
    2       $TITLE =  T E S T   M I S S I L E
    3       $NASTRAN = YES
    4       $PLOTID =  MCKEE, CODE1844, EXT71493
    5       $PUNCH = YES
    6        $ ONE QUAD MUST BE REPLACED BY A TRIANGLE AT THE ROOT OF EACH FIN.
    7        $END
    8        SOL 2,0
    9        TIME 10
   10        APP DISP
   11        CEND
   12        TITLE = TEST MISSILE
   13        BEGIN BULK
```

CARD COUNT INPUT DATA DECK

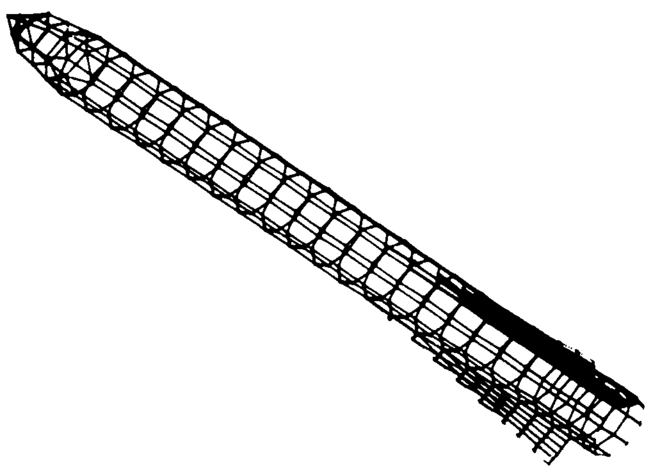
1	IEQU	40	2	40	1	40	4	40	3
2	IEQU	40	6	40	5				
3	MAT1	1	1.E7		.27	1.E-5			
4	CONEND	10	1	20	3			1.	120.+0101
5	+0101	.67	.02		2				+0102
6	+0102		.02		2				
7	CONEND	10	3	20	5				120.+0201
8	+0201		.02		2				+0202
9	+0202		.02						
10	CONEND	10	5	20	1				120.+0301
11	+0301		.02		2				+0302
12	+0302		2			1			
13	CONE	20	1	30	3			2.0	+0401
14	+0401		.025						+0402
15	+0402	1.0			3				
16	CONE	20	3	30	5				+0501
17	+0501		.025						
18	CONE	20	5	30	1				120.0+0601
19	+0601		.025						
20	CONE	30	1	40	3			12.0	+0701
21	+0701		.025		3				+0702
22	+0702	1.0			12				
23	CONE	30	3	40	5				+0801
24	+0801		.025		3				
25	CONE	30	5	40	1				120.0+0901
26	+0901		.025		3				
27	CONE	40	1	50	3			6.0	+1001
28	+1001		.03						+1002
29	+1002	1.0			6				
30	CONE	40	3	50	5				+1101
31	+1101		.03						
32	CONE	40	5	50	1				120.0+1201
33	+1201		.03						
34	CONE	50	1	60	3			2.0	+1301
35	+1301		.03						+1302
36	+1302	1.0			3				
37	CONE	50	3	60	5				+1401
38	+1401		.03						
39	CONE	50	5	60	1				120.0+1501
40	+1501		.03						
41	CONE	60	1	70	3			1.0	+1601
42	+1601		.035						.1602
43	+1602	1.0			2				
44	CONE	60	3	70	5				+1701
45	+1701		.035						
46	CONE	60	5	70	1				120.0+1801
47	+1801		.035						
48	CONE	40	2	50	1			6.0	+1901
49	+1901		.04		1				+1902
50	+1902	1.75			6				

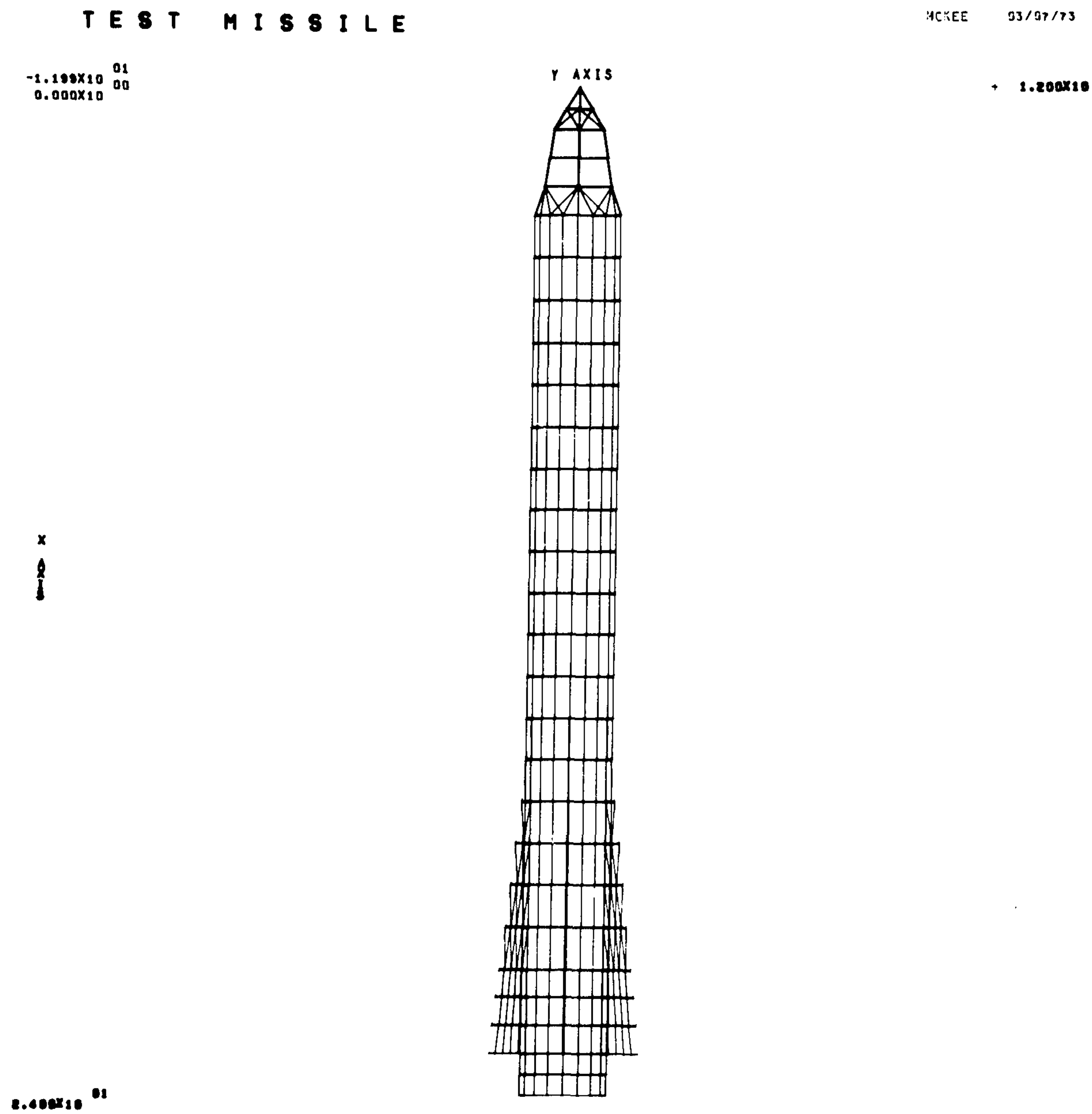
INPUT DATA DECK

CARD COUNT							
51	CONE	40	4	50	3		
52	+2001		.04		1	6.0	+2001
53	+2002	1.75			6		+2002
54	CONE	40	6	50	5		
55	+2101		.04		1	6.0	+2101
56	+2102	1.75			6		+2102
57	CONE	50	2	60	1		
58	+2201		.04		5	2.0	+2201
59	+2202	2.0			3		+2202
60	CONE	50	4	60	3		
61	+2301		.04		5	2.0	+2301
62	+2302	2.0			3		+2302
63	CONE	50	6	60	5		
64	+2401		.04		5	2.0	+2401
65	+2402	2.0			3		+2402
66	ENDDATA						

TEST MISSILE

PERSPECTIVE PLOT




TEST MISSILE
MCKEE 03/07/73
-1.199X10 01
0.000X10 00
Y AXIS
+ 1.200X10
X AXIS
2.400X10 01

ACKNOWLEDGMENTS

The authors wish to thank the many people who have contributed to the development of this program through their comments and suggestions. In particular, we wish to thank Dr. E.H. Cuthill for many stimulating conversations and her cheerful encouragement throughout the project. We wish to express our appreciation for the use of subroutines developed by Dr. Gordon C. Everstine (GOOGAN), Mrs. Barbara Kelly (NASPLT), and for consultation and programming assistance from Mr. Richard Kazden (SETUP) and Mr. Michael Golden. We are also indebted to the developers of the NASTRAN program for many of the data handling techniques and user-oriented data formats used in this program.

INITIAL DISTRIBUTION

Copies	
2	DIA, Washington, D.C.
	1 K. Goering
	1 R. Wood
1	U.S. Army Elec Command
	1 A.L. Sigismondi
1	U.S. Army Mobility Equip R&D
	1 J. Marburger
4	U.S. Army Harry Diamond Labs
	1 G.J. Hutchins
	1 P.J. Emmerman
	1 R. Hamilton
	1 C. Anstine, Jr.
3	U.S. Army Frankford Arsenal
	1 D.L. Frederick
	1 P.F. Gordon
	1 S.H. Eisman
2	U.S. Army Picatinny Arsenal
	1 W. Bolte
	1 B. Nagel
2	U.S. Army Watervliet Arsenal
	1 P.C.T. Chen
	1 G.P. O'Hara
1	CNO (OP-98)
4	CHONR
	1 N. Basdekas
	1 N. Perrone
	1 S. Brodsky
	1 R.J. Lundegard
1	ONR, Chicago
	1 R.N. Buchal
2	NRL
	1 R. Perlut
	1 Lib
1	CHNAVMAT
	1 S. Atchison
1	DNL

Copies	
3	USNA
	1 Dept of Math
	1 Aerospace Dept
	1 Tech Lib
1	NAVPGSCOL
1	NROTC
1	NAVWARCOL
3	NAVSHIPSYSCOM
	1 Dr. J.H. Huth
	1 R.R. Kolesar
	1 Tech Lib
2	NAVAIRSYSCOM
	1 G.P. Maggos
	1 Tech Lib
4	NAVFACENGCOM
	1 M. Yachnis
	1 H.D. Nickerson
	1 M.B. Hermann
	1 Tech Lib
3	NAVORDSYSCOM
	1 O. Seidman
	1 L. Pasiuk
	1 Lib
1	NAVOCEANO
	Tech Lib
1	NAVTRAEQUIPCEN
	Tech Lib
3	NAVAIRDEVCEN
	1 A. Somoroff
	1 S.L. Huang
	1 Lib
2	NELC
	1 E. Nissan
	1 Tech Lib

Copies

5 NUC San Diego
  1 J.T. Hung
  1 L. McCleary
  1 D. Barach
  1 G. Benthien
  1 Lib

3 NUC Pasadena
  1 C.F. Falkenback
  1 P.D. Burke
  1 Lib

2 NUC Hawaii
  1 A.T. Strickland
  1 Lib

4 NWC
  1 J. Serpanos
  1 D.E. Zilmer
  1 W.J. Stronge
  1 Tech Lib

2 NCEL
  1 J. Crawford
  1 Lib

1 NSSNF
  Tech Lib

3 NOL
  1 R.J. Edwards
  1 P.C. Huang
  1 Tech Lib

4 NWL
  1 C.M. Blackmon
  1 J. Schwartz
  1 P. Johnson
  1 Tech Lib

1 NUSC NPT

4 NUSC NLON
  1 A. Carlson
  1 R. Manstan
  1 J.W. Frye
  1 R. Dunham

1 NAVSHIPYD BREM

Copies

1 NAVSHIPYD BSN

2 NAVSHIPYD CHASN
  1 J.B. Kruse

3 NAVSHIPYD MARE ISLAND
  1 D. Mitchell
  1 R. Munden

1 NAVSHIPYD NORVA

1 NAVSHIPYD PEARL

1 NAVSHIPYD PHILA

2 NAVSHIPYD PTSMH
  1 E.A. Fernald

6 NAVSEC
  1 N. Griest
  1 R.P. Lavoie
  1 G.J. Snyder
  1 T.N. Hull
  1 R.J. Keltie
  1 Tech Lib

1 NAVWPNSENGSUPPACT

1 NAVSPASYSACT

1 NAVAERORECOVFAC

1 NAVAVIONICFAC, Indianapolis
  1 S. Stephen

1 PACMISTRAN

1 NAVAIRTESTCEN

1 NAVAIRTESTFAC

2 NAVAIRENGCEN
  1 T. Mazella
  1 Tech Lib

1 NAVAIRPROPTESTCEN

1 NAVWPNEVALFAC

1 NAVEODFAC

1 NAVORDTESTU

1 NAVORDMISTESTFAC

Copies	
1	NAVMISCEN
4	U.S. Air Force Flight Dynamics Lab
	1 J. Johnson
	1 R. Taylor
	1 F. Janik
	1 Lib
1	U.S. Air Force Flight Propulsion Lab
	1 LT James MacBain
12	DDC
5	COGARD
	1 W.A. Cleary
	1 R. Johnson
	1 LCDR V.F. Kieth
	1 LTjg M.C. Grosteck
	1 LT J.C. Card
1	NASA HQS
	1 D. Michel
1	NASA Ames
	1 P. Polentz
1	NASA Flight Res Cen
	1 A. Carter
3	NASA Goddard
	1 T. Butler
	1 W. Case
	1 J. Mason
1	NASA Kennedy
	1 H. Harris
3	NASA Langley
	1 R. Butler
	1 R. Fulton
	1 J.P. Raney
1	NASA Lewis Lib
1	NASA Manned Spacecraft Ctr
	1 W. Renegar

Copies	
1	NASA Marshall
	1 R. McComas
1	Calif Inst of Technology Jet Propulsion Lab
	1 Dr. Roy Levy
4	Johns Hopkins Univ, APL
	1 W. Caywood
	1 G. Dailey
	1 R.M. Rivello
	1 J.S. O'Connor
2	Penn State, ORL
	1 J.C. Conway
	1 Lib
1	Univ of Virginia, Civil Engineering Dept
	1 Dr. Richard Eppink
1	Mrs. Anne Woehr Advanced Structural Analysis (C-6), Bell Aerospace Co, P.O. Box 1 Buffalo, New York 14240
1	Mr. R.B. Smith, Manager Engineering Design Prog. Bettis Atomic Power Lab. Box 79 West Mifflin, Pa. 15122
1	Dr. R.H. MacNeal MacNeal-Schwendler Corp. 7442 N. Figueroa Street Los Angeles, Calif 90041
1	Mr. Jack Conaway Sikorsky Aircraft Structures & Materials R&D North Main Street Stratford, Conn. 06497

CENTER DISTRIBUTION

Copies	Code	Copies	Code
1	01	1	1853
1	11 (W.M. Ellsworth)	1	1854
1	15 (Dr. W.E. Cummins)	1	186
1	16 (Dr. H.R. Chaplin)	1	188
1	17 (Dr. W.W. Murray)	1	19 (P.S. Dell Aria)
1	172	1	194
1	1721	1	196
1	1725	1	1962
1	1727	1	27 (H.V. Nutt)
1	173	1	271
1	1731	1	272
1	1735	2	2723 (Coblenz, Rajan)
1	174	1	273
1	1745	1	2731
1	177 (UERD PTSMH)	1	274
1	177B (UERD PTSMH)	1	2741
1	1775 (UERD PTSMH)	1	2742
1	18 (G. Gleissner)	1	275
1	1805	1	276
1	1808/1809	1	28
1	183	1	287
1	184	1	94 (J.L. Decker)
1	1842	1	9424 (M.E. Lunchick)

☆U.S. GOVERNMENT PRINTING OFFICE: 1973 542-168/169 1-3

UNCLASSIFIED

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY *(Corporate author)* Naval Ship Research and Development Center Bethesda, Maryland 20034	2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED 2b. GROUP

3. REPORT TITLE

A General Purpose Data Generator for Finite Element Analysis

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

Final

5. AUTHOR(S) *(First name, middle initial, last name)*

James M. McKee
Evangeline T. Marcus

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
April 1973	191	3

8a. CONTRACT OR GRANT NO. b. PROJECT NO. SR 535 32 106, Task 15326 c. Work Unit 1-1844-916 d.	9a. ORIGINATOR'S REPORT NUMBER(S) 4066 9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*

10. DISTRIBUTION STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	NAVSHIPS (0311)

13. ABSTRACT

A computer program system has been developed to automate the preparation of a finite element model to be analyzed using the NASTRAN general purpose structural analysis program. Using engineering conventions and modular "building block" specifications, the program minimizes both the manual effort and the probability of an undetected error in the preparation of NASTRAN data.

This document is intended to be both a guide for the user of the program and a programmer's reference for the modification and further development of the program.

DD FORM 1 NOV 65 1473 (PAGE 1)

S/N 0101-807-6801

UNCLASSIFIED

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Finite Element Analysis						
Data Generation						
NASTRAN						
Structural Analysis - Computer Methods						
Finite Element Idealization						